Framework para la Evaluación Automática de la Calidad de los Requerimientos de Software basado en Redes Neuronales

María Gramajo

TESIS DOCTORAL

DE LA UNIVERSIDAD TECNOLÓGICA NACIONAL

Y LA UNIVERSITÉ DE TECHNOLOGIE DE BELFORT MONTBÉLIARD

Doctorado en Ingeniería

Mención en Sistemas de Información

por

Maria Guadalupe Gramajo

Framework para la Evaluación Automática de la Calidad de los Requerimientos de Software basado en Redes Neuronales

Esta Tesis fue presentada en Santa Fe, Argentina, el 28 de Septiembre de 2023

Composición del Jurado :

Dr. Marciszak Marcelo	Profesor en la Universidad Tecnológica Nacional	Presidente
Dra. Hadad Graciela	Profesora en la Universidad Nacional del Oeste	Examinadora
Dra. Arévalo Gabriela	Profesora en la Universidad Nacional de Quilmes	Examinadora
Dr. Steffenel Luiz	Profesor en la Université de Reims Champagne-Ardenne	Examinador
Dra. Ballejos Luciana	Profesora en la Universidad Tecnológica Nacional	Directora
Dra. Ale Mariel	Profesora en la Universidad Tecnológica Nacional	Codirectora
Dra. Baala Oumaya	Profesora en la Université de Technologie de Belfort Montbéliard	Codirectora

N°

Agradecimientos

Me gustaría expresar mi más sincero agradecimiento a mis directoras, Luciana y Mariel. Su guía, paciencia y dedicación han sido una inspiración constante a lo largo de este arduo proceso. Ambas me han ayudado a crecer tanto personal como profesionalmente.

Asimismo quiero aprovechar esta oportunidad para agradecer a Oumaya, quien ha sido mi mentora durante mi estadía en Francia. Gracias por su apoyo, orientación y confianza en mis capacidades.

También quiero extender mi agradecimiento a Álvaro, quien siempre estuvo presente para brindar su apoyo incondicional y palabras de aliento. Su respaldo constante y su disposición para ayudarme en los momentos más desafiantes han sido un gran motor que me ha impulsado a superar obstáculos y a perseguir mis metas con determinación.

Del mismo modo quisiera expresar mi agradecimiento a todas las personas que con sus palabras de aliento y consejos, han contribuido a mi desarrollo académico y personal a lo largo de estos años: Baptista, Amine, Santiago, Rusber y Julia.

Por último, quiero agradecer a mis hermanos Bruno, Macarena y Benjamín, quienes han estado a mi lado durante todo este proceso, brindándome su amor, comprensión y aliento.

Y, finalmente al Consejo Nacional de Investigaciones Científicas y Técnicas por haberme otorgado la beca que me ha permitido continuar con mis estudios de doctorado.

Nuevamente, gracias a todos los que han formado parte de esta travesía académica y han contribuido a la culminación de mi tesis doctoral.

Índice

1

Introducción

En este capítulo se describe el contexto en el cual se enmarca el presente trabajo de Tesis. Además, se presentan los antecedentes y las motivaciones que dieron origen a la propuesta, junto con el objetivo general y los aportes realizados. Por último, la organización del contenido de la Tesis también se incluye en este capítulo.

1.1/ Contexto

La Ingeniería de Software se encuentra en una búsqueda constante de técnicas de soporte más eficientes para el desarrollo de productos de software. En este sentido, la comunidad académica ha orientado sus esfuerzos a través de los años hacia la definición de nuevos métodos y técnicas que contribuyan al desarrollo de software de calidad. El término calidad es ampliamente reconocido por su naturaleza compleja, subjetiva y polifacética (Kitchenham y Pfleeger, 1996). En efecto, la calidad del software ha resultado uno de los propósitos más pretendidos en el área. Especialmente, si se considera que hoy en día los sistemas de información se han convertido en un activo esencial en todas las organizaciones. En virtud de ello, los productos de software se construyen satisfaciendo requerimientos más complejos, debido a las exigencias y características críticas que demanda la competencia global.

Desde una perspectiva de la Ingeniería de Requerimientos (IR), diversas investigaciones han demostrado que alcanzar la calidad de los requerimientos es el primer paso hacia el desarrollo de software de calidad (Fabbrini et al., 2001b; Kamata y Tamai, 2007; Knauss et al., 2016; **?**). De hecho, en éstas también se señala que la calidad de los requerimientos constituye un factor determinante de éxito para un proyecto de software. La razón principal se debe a que éstos son el contenido principal de la Especificación de Requerimientos de Software (ERS), fundamental y transversal al proceso de desarrollo de software.

En relación a esto, el lenguaje natural se ha posicionado como una de las técnicas más utilizadas para obtener y documentar los requerimientos (Denger et al., 2003; Femmer, 2013; Dalpiaz et al., 2018). Aunque a menudo se complementan con modelos y notaciones con diversos grados de formalidad y detalle, los requerimientos continúan siendo especificados en

lenguaje natural en la industria de software. Esta tendencia se asocia a la flexibilidad y la comprensibilidad que proporciona el lenguaje natural, que se caracteriza por ser un medio de comunicación expresivo, intuitivo y universal (Sommerville, 2020). Estos aspectos respaldan su aplicación para comunicar y documentar las necesidades de los interesados de forma práctica y rápida (Kocerka et al., 2018).

Sin embargo, diversos problemas relacionados a la calidad de los requerimientos surgen a causa de la flexibilidad y la naturaleza inherente que proporciona el lenguaje natural. Desafortunadamente, su uso conduce a la especificación de requerimientos propensos a inconsistencias, redundancias y ambigüedades. En consecuencia, si esta situación no se aborda desde fases tempranas, puede impactar de manera negativa en las fases posteriores del ciclo de vida del software (Denger y Olsson, 2005). En particular, los costos asociados a la corrección de errores y defectos que surgen a causa de requerimientos de baja calidad -considerados los más difíciles de corregir- resultan críticos. Esto se debe principalmente a que su corrección puede requerir cambios significativos en el diseño y la arquitectura del sistema, lo que implica un mayor riesgo de retrasos en el proyecto y, por consiguiente, la posibilidad de cancelación.

El aseguramiento de calidad de los requerimientos ha sido estudiado desde varias perspectivas. En tal sentido, se han desarrollado diversas propuestas para abordar desafíos relacionados con la calidad de los requerimientos. Entre ellas se destacan los sistemas basados en reglas, modelado y formalización, y clasificación automática (Kummler, 2021).

La mayoría de las propuestas coinciden en el uso de técnicas de procesamiento de lenguaje natural para obtener secuencias representativas de los requerimientos e implementar mecanismos de control. Entre ellos, reglas gramaticales, expresiones regulares, corpus de palabras y/o frases. Luego, las representaciones de requerimientos se procesan mediante la ejecución de reglas y sentencias condicionales definidas a priori, con el propósito de obtener indicadores cuantitativos de calidad.

Sin embargo, este tipo de estrategias conduce a procesos de evaluación estáticos de la calidad, dado que en su mayoría no son capaces de extraer información contextual. Por tanto, en estos casos, la determinación de la calidad se lleva a cabo bajo *contextos controlados*. De hecho, algunas estrategias utilizadas para abordar la calidad de los requerimientos pueden considerarse obsoletas, o bien, de rendimiento cuestionable respecto a las exigencias y características críticas que surgen del contexto actual (tiempo, costos, complejidad, volumen de datos). Es por ello que resulta clave la integración de nuevas tecnologías para enriquecer procedimientos tradicionales que contribuyan al estudio de la calidad.

En este sentido, la aplicación de técnicas de Inteligencia Artificial (IA) ha demostrado no ser ajena a la IR. Durante los últimos años, la IA se ha posicionado como una herramienta poderosa y accesible, capaz de ser utilizada como un componente clave en el desarrollo de productos de software. La integración de estas tecnologías tiene por objetivo optimizar el proceso de desarrollo y automatizar tareas a fin de obtener sistemas de calidad (Gramajo et al., 2020a).

En este contexto, la aplicación de técnicas de aprendizaje que derivan de la IA tales como el procesamiento del lenguaje natural, aprendizaje automático, redes neuronales, entre otras, se ha incrementado exponencialmente. Este éxito se debe principalmente a su capacidad para abordar problemas complejos a través de múltiples niveles de representación y abstracción, lo que facilita el procesamiento y el análisis de datos de series temporales, texto, sonido e imágenes (Young et al., 2018).

En particular, la aplicación de redes neuronales ha sido predominante debido a su capacidad para procesar información secuencial. Las *redes neuronales recurrentes* se consideran adecuadas para modelar las dependencias de contexto presentes en el lenguaje y tareas relacionadas con el modelado de secuencias (Mikolov et al., 2010). Por su parte, las *redes neuronales transformadoras* ofrecen nuevos mecanismos de autoatención y codificadores posicionales que ayudan a capturar las relaciones contextuales entre palabras, a pesar de la distancia temporal (Brașoveanu y Andonie, 2020). Ambos tipos de redes neuronales poseen los mecanismos suficientes para tratar la naturaleza secuencial del lenguaje y la dependencia contextual. A causa de ello, permiten abordar el procesamiento y análisis de los requerimientos de software expresados en lenguaje natural y, a partir de esto, evaluar su calidad.

Por todo lo expuesto, en este trabajo de Tesis se presenta un conjunto de modelos de clasificación binaria basados en redes neuronales para evaluar las características de calidad de requerimientos de software individuales. Esta propuesta tiene por objetivo proporcionar mecanismos automáticos de control y verificación de la calidad de los requerimientos de software a ingenieros de requerimientos y profesionales del software.

Las características de calidad abordadas se establecen en el estándar vigente ISO IEC IEEE 29148 (2018). El conjunto de clasificadores adopta un enfoque de aprendizaje del tipo supervisado. Con el propósito de proveer los datos necesarios para su entrenamiento, se propone en esta tesis la construcción de un corpus de requerimientos que cubre múltiples dominios procedentes de diversas compañías y proyectos universitarios. Esto último contribuye a la heterogeneidad de los datos considerados y, con ello, a la generalización de las predicciones sobre nuevos requerimientos.

En esta investigación también se define un método que permite la extracción de características de los requerimientos de software expresados en lenguaje natural, para estructurarlos en un formato de entrada adecuado para los modelos neuronales. Además, se describe la configuración experimental que permite la ejecución de los modelos neuronales. Esto último comprende la búsqueda de los valores paramétricos óptimos para la obtención de aquellos modelos neuronales con mejor desempeño. Los resultados se reportan en base a los valores proporcionados por medidas de desempeño ampliamente aceptadas por la comunidad académica. Del mismo modo, se ofrece un análisis comparativo de los resultados obtenidos respecto a las variantes de redes neuronales: Long Short-Term Memory (LSTM), Bidirectional Long Short Term Memory (BiLSTM), Gated Recurrent Units (GRU), Bidirectional Gated Recurrent Units (BiGRU) y Bidirectional Encoder Representations from Transformers (BERT) contempladas

en esta investigación.

Por último, la solución propuesta es integrada a un framework que permite la evaluación automática de la calidad de los requerimientos. Dicho framework define un flujo de trabajo compuesto por cuatro simples pasos que facilitan las interacciones entre el ingeniero y la herramienta, a través de interfaces de usuario. La herramienta computacional desarrollada provee los mecanismos necesarios para la carga de requerimientos que se desean evaluar, realizar ajustes adicionales sobre los requerimientos y evaluar los requerimientos previamente cargados mediante la ejecución del conjunto de clasificadores basados en redes neuronales profundas. Cabe destacar que estos últimos han sido integrados como servicios en la herramienta para proporcionar las funcionalidades de evaluación automática. Por último, el ingeniero tiene la opción de realizar correcciones sobre los resultados que retorna la herramienta si éstos no satisfacen su criterio. Finalmente, la herramienta proporciona las opciones de generación de reportes, los cuales permiten visualizar e interpretar los resultados obtenidos de forma rápida y fácil.

1.2/ Objetivos

El objetivo general de esta Tesis Doctoral es el desarrollo de un framework para la evaluación automática de la calidad de los requerimientos de software. Dicha herramienta, integra técnicas de procesamiento del lenguaje natural y aprendizaje profundo. La construcción de la herramienta de soporte al proceso de evaluación de la calidad requiere alcanzar los siguientes objetivos específicos:

- Establecer el conjunto de características de calidad que serán evaluadas de forma automática.
- Definir métodos de preprocesamiento que permitan estructurar los requerimientos en un formato de entrada adecuado para los modelos neuronales.
- Construir un conjunto de clasificadores basado en redes neuronales profundas. Para ello se debe considerar:
  - Requerimientos de los modelos neuronales;
  - Recolección de datos y extracción de características;
  - Entrenamiento y prueba de los modelos neuronales.
- Analizar y evaluar el desempeño de los modelos neuronales mediante el cálculo de métricas definidas en la literatura.
- Analizar comparativamente el desempeño de cada variante de red neuronal, a fin de determinar la más apropiada para abordar la evaluación de la calidad de los requerimientos.

- Implementar e integrar los modelos neuronales en una herramienta computacional.
- Desarrollar un caso de estudio práctico para validar el enfoque propuesto.
- Proponer lineamientos para la adopción del enfoque y herramienta propuesta en proyectos de desarrollo de productos de software, particularmente en el proceso de la IR.

1.3/ Principales Contribuciones

Este trabajo de Tesis focaliza en el estudio sobre las aplicaciones de técnicas de aprendizaje profundo y procesamiento de lenguaje natural para desarrollar clasificadores que permitan determinar la calidad de los requerimientos de software expresados en lenguaje natural. Para ello, se tomaron como referencia las características de calidad definidas en el estándar ISO IEC IEEE 29148 (2018).

Inicialmente, con el propósito de indagar sobre las posibles aplicaciones de estas técnicas en el área de la IR, se ha llevado a cabo una revisión exploratoria de la literatura. Para dicha investigación se definió como objetivo identificar y analizar tendencias, conjuntos de datos y algoritmos de aprendizaje automático utilizados en la resolución de tareas y desafíos en dicha área. Estos aportes fueron oportunamente publicados en el IV Congreso Bienal de la Sección Argentina del IEEE (Gramajo et al., 2018) y, posteriormente, extendido para ser publicado en IEEE Latin America Transactions Journal (Gramajo et al., 2020a).

El análisis de los estudios seleccionados, demostró que las técnicas con un enfoque de aprendizaje del tipo supervisado pueden aplicarse efectivamente para resolver y automatizar tareas en la IR. Principalmente, los estudios se centraron en la detección de problemas lingüísticos en documentos de requerimientos y artefactos escritos en lenguaje natural, además de la clasificación y trazabilidad de los requerimientos. Los resultados obtenidos en este estudio inicial permitieron evidenciar el valor de los algoritmos de aprendizaje automático en el área. Principalmente, se destaca la posibilidad que brindan estos algoritmos de optimizar y automatizar tareas que demandan grandes esfuerzos. Por otro lado, cabe resaltar que no se encontraron trabajos que desestimen la aplicación de estas técnicas en la obtención de mejoras y resolución de problemas.

Posteriormente se establecieron criterios para determinar las características de calidad definidas por el estándar ISO IEC IEEE 29148 (2018) que son factibles de ser evaluadas a través de la aplicación de redes neuronales profundas y técnicas de procesamiento de lenguaje natural. En base a ello, el conjunto de características de calidad bajo estudio se ha conformado por las características de calidad *Completo*, *Correcto*, *Singular*, *Factible*, *Verificable*, *Conforme*, *No ambiguo* y *Apropiado*.

Luego, se llevó a cabo un estudio preliminar que aborda en primera instancia la característica de calidad singular, siendo esta última parte del conjunto de características seleccionadas.

Dicho estudio incluyó la definición de un modelo neuronal que utiliza redes neuronales recurrentes del tipo LSTM y GRU para evaluar la singularidad de los requerimientos de software. Cabe mencionar que para llevar a cabo el entrenamiento de dichos modelos se ha construido un corpus de requerimientos expresados en lenguaje natural. También, se definió un método que permite la extracción de características de los requerimientos, a fin de estructurarlos en un formato de entrada adecuado para los modelos neuronales.

Los resultados obtenidos en este primer estudio demostraron la capacidad de las redes neuronales profundas para abordar este tópico de investigación. Este último, ha sido publicado en el Workshop on Requirement Engineering (Gramajo et al., 2020b). En efecto, motivado por los resultados obtenidos la investigación, se extendió a la evaluación de las características de calidad restantes (Gramajo et al., 2021).

En este sentido, se destaca que en esta investigación se consideraron múltiples variantes de redes neuronales para definir diversos modelos neuronales por cada característica de calidad bajo estudio, entre ellas se encuentran las arquitecturas LSTM, BiLSTM, GRU, BiGRU y BERT. Cada modelo neuronal obtenido ha sido sometido a un proceso de entrenamiento y evaluación. Además, también se provee un análisis comparativo respecto al desempeño de cada modelo neuronal.

Por último, la solución propuesta ha sido integrada en una herramienta computacional que tiene el propósito de proveer los mecanismos necesarios para llevar a cabo la evaluación automática de la calidad de los requerimientos y, con ello, asistir a los ingenieros en las tareas de verificación. Adicionalmente, se presenta la documentación correspondiente a los artefactos de análisis y diseño de la herramienta computacional. A continuación, la siguiente sección describe de qué manera se organiza el contenido descrito en este trabajo de Tesis.

1.4/ Organización de la Tesis

El resto del contenido de este trabajo de Tesis se encuentra organizado de la siguiente manera.

El *Capítulo 2* introduce el concepto de calidad en la IR. Luego, se describen las características de calidad para requerimientos individuales incluidas en el estándar ISO IEC IEEE 29148 (2018). Posteriormente, se discute la integración de las tecnologías de la IA en el área de la IR a través del análisis de los resultados obtenidos en la revisión de la literatura descrita en la sección anterior.

El *Capítulo 3* describe la metodología utilizada para obtener los modelos neuronales. Del mismo modo, se presenta el análisis descriptivo del corpus de requerimientos generado para entrenar cada modelo neuronal. También se incluye la configuración experimental de los modelos neuronales (parámetros e hiperparámetros). Por último, el análisis comparativo respecto al desempeño de cada variante de red neuronal considerada, también se detalla en este capítulo.

El *Capítulo 4* presenta el desarrollo de una herramienta web denominada *Quality Assurance*

Support System que integra los modelos neuronales definidos como servicios para asistir a los ingenieros en la evaluación automática de la calidad de los requerimientos. Del mismo modo, en este capítulo se discute sobre la inserción de la herramienta propuesta en el proceso de la IR.

En el *Capítulo 5*, con el propósito de validar la propuesta, se incluye un caso de estudio asociado a un sistema que provee servicios bancarios. A través de la descripción de las interacciones entre el ingeniero y la herramienta, se demuestra la factibilidad de uso de la propuesta en el proceso de la IR en las tareas de verificación.

Por último, en el *Capítulo 6* se presentan las conclusiones y futuras líneas de investigación.

1.5/ Summary

This chapter describes the context in which this Thesis is presented. Then, the background and motivations that originated the proposal are presented. In addition, the general objective and the contributions accomplished are described. Finally, the organization of the content is also included in this chapter.

1.5.1/ Goals

The main purpose of this Thesis is the development of a framework for the automatic evaluation of the quality of software requirements. This tool integrates natural language processing and deep learning techniques.

The development of the tool to support the quality assessment process requires achieving the following specific goals:

- Define the set of quality characteristics that will be evaluated automatically.
- Establish preprocessing methods that allow structuring the requirements in a suitable input format for the neural models.
- Development a set of binary classifiers based on deep neural networks this includes:
  - Neural model requirements;
  - Data collection and feature extraction;
  - Training and testing of neural models.
- Analyze and evaluate the performance of the neural models through metrics defined in the literature.
- To comparatively analyze the performance of each neural network variant in order to determine the most appropriate for the study of the quality of the software requirements.

- Implement and integrate neural models in a computational tool. Develop a proof of concept to validate the proposed approach.

- Define guidelines for the implementation of the proposed approach and tool in software product development projects.

1.5.2/ Main Contributions

This Thesis focuses mainly on the study of the applications of deep learning and natural language processing techniques to develop classifiers to determine the quality of software requirements expressed in natural language. For this purpose, we have taken as a reference the quality characteristics defined in the standard ISO IEC IEEE 29148 (2018). Initially, in order to investigate the possible applications of these techniques in the area of Requirement Engineering, an exploratory review of the literature was performed.

The research aims to identify and analyze trends, datasets, and machine learning algorithms used in solving tasks and challenges in this area. These contributions were opportunely published in the IV Biennial Congress of the IEEE Argentina Section (Gramajo, Ballejos and Ale, 2018) and, subsequently, extended for publication in IEEE Latin America Transactions Journal (Gramajo et al., 2020a).

The analysis of the selected studies showed that techniques with a supervised learning approach can be effectively applied to solve and automate tasks in Requirement Engineering. Mainly, the studies focused on the detection of linguistic problems in requirements documents and artifacts written in natural language. In addition to the classification and traceability of requirements.

The results obtained in this initial study made it possible to demonstrate the value of machine learning algorithms in this area. Mainly, it is highlighted that they allow us to optimize and automate tasks that require intensive efforts. On the other hand, it is important to mention that no studies were found that dismissed the application of these techniques in obtaining improvements and solving problems.

Subsequently, the quality characteristics provided in the ISO IEC IEEE 29148 (2018) the standard was analyzed from a technical perspective to determine those candidates to be evaluated through the application of deep neural networks.

Then, a preliminary study has been performed including the definition of neural models using recurrent neural networks of LSTM and GRU type to evaluate the singular quality characteristic. It is important to mention that a corpus of requirements has been built to train these models. Furthermore, a method has been defined that allows the extraction of features from software requirements expressed in natural language.

The results obtained in this preliminary study demonstrated the ability of deep neural networks to address this research topic. The study has been published in the Workshop on

Requirement Engineering (Gramajo et al., 2020b).

Subsequently, based on the results obtained, the research was extended to the evaluation of the rest of the quality characteristics (Gramajo et al., 2021).

Then, the defined neural models have been subjected to a training and evaluation process. Multiple variants of neural networks have been addressed in this research. Long Short Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Gated Recurrent Unit (GRU), Bidirectional Recurrent Unit (BiGRU), and, in addition, Bidirectional Encoder Representations from Transformers (BERT). Therefore, a comparative analysis regarding the performance of each neural model is also provided. Finally, the proposed solution is integrated into a computational tool that has the purpose of providing the necessary mechanisms to carry out the automatic evaluation of the quality of the requirements and thus, assist the engineers in the verification tasks. Additionally, the documentation corresponding to the analysis and design artifacts of the proposed tool is presented.

2

Contexto y Problema

Este capítulo introduce el concepto de calidad en la Ingeniería de Requerimientos. Para lograrlo, se presentan los fundamentos y aspectos relevantes que conciernen a las características de calidad para requerimientos de software individuales, incluidas en el estándar ISO IEC IEEE 29148 (2018) que es el documento base para el desarrollo de esta investigación. Luego, se mencionan los desafíos asociados al aseguramiento de la calidad de los requerimientos expresados en lenguaje natural dentro de un proyecto de software. Posteriormente, se analiza a través de una revisión sistemática de la literatura, la incorporación de las tecnologías de la Inteligencia Artificial en la Ingeniería de Requerimientos como un medio exploratorio para dar respuesta a los problemas emergentes en dicha área. En particular, se aborda la aplicación de estas tecnologías en la evaluación de la calidad de los requerimientos de software.

2.1/ Calidad en la Ingeniería de Requerimientos

La calidad es un término ampliamente discutido en la Ingeniería de Software, reconocido por su naturaleza compleja, subjetiva y polifacética (Kitchenham y Pfleeger, 1996; Sommerville, 2011). En general, la calidad de un producto o servicio depende siempre de la percepción y expectativas del receptor o cliente. Desafortunadamente, la literatura carece de una definición unívoca del término calidad. Aun así, desarrollar productos de software de calidad es una necesidad consensuada por los líderes de proyecto e ingenieros de software y resulta uno de los propósitos más pretendidos en la industria del software (Lindgren et al., 2008). Consecuentemente, esto ha dado lugar a numerosos esfuerzos de diversos autores para proporcionar una interpretación asequible del término calidad y establecer métodos para su evaluación.

De acuerdo con el estándar IEEE 1061 (1998), la calidad del software se define como el grado en el cual un producto de software posee una combinación de atributos deseados. Dicha combinación de atributos debe ser claramente definida, de lo contrario, la evaluación de la calidad queda sujeta a la intuición.

Este estándar propone concebir la calidad de un producto de software como la definición y evaluación de un conjunto de atributos de calidad requeridos para un software particular. Luego,

para medir los atributos de calidad se identifica un conjunto apropiado de métricas de software. El propósito de las métricas es hacer evaluaciones a lo largo del ciclo de vida del software a fin de determinar el cumplimiento de los atributos de calidad. El uso de métricas reduce la subjetividad en la evaluación y control de la calidad del software, además de proporcionar una base cuantitativa para la toma de decisiones.

Sin embargo, son muchos los factores que influyen en la calidad de un producto de software y, por lo tanto, además de evaluar y supervisar directamente la calidad del software, es necesario analizar los procesos utilizados para su desarrollo (Hsia et al., 1993). En relación con esto, diversas investigaciones reconocen que un producto de software es tan bueno tal y como lo es su proceso de desarrollo (Ashrafi, 2003; Wagner, 2013). Dicho de otro modo, los procesos de desarrollo de calidad conducen a productos de software de calidad. La correcta comprensión y especificación de los requerimientos es parte integral de estos procesos. La disciplina que proporciona el mecanismo adecuado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida que se transforman en un sistema funcional, es la Ingeniería de Requerimientos (IR).

Desde la perspectiva de la IR, Femmer y Vogelsang (2018) consideran que la calidad está asociada a los siguientes objetivos:

- *Comprensión de las necesidades de las partes interesadas*, esto incluye la interpretación correcta y completa de los objetivos, expectativas y necesidades de los interesados en el sistema.

- *Definición de un acuerdo*, que manifieste el consenso de todas las partes interesadas respecto al sistema a desarrollar. Para lograr tal acuerdo, los requerimientos deben priorizarse correctamente y, además, se debe garantizar a los interesados que se desarrollará la mejor solución posible, sujeta a las capacidades técnicas del equipo de trabajo, para cubrir sus necesidades.

- *Establecer un marco de referencia respecto al sistema a desarrollar*. Esto consiste en transmitir de forma correcta y completa las necesidades de las partes interesadas, así como, el acuerdo establecido, a todos los integrantes del proyecto, a fin de lograr una interpretación y percepción común del sistema a desarrollar.

- *Gestionar las actividades del proceso de Ingeniería de Requerimientos* de forma eficiente y eficaz. Dicho proceso, involucra las actividades de obtención, análisis, especificación y validación de los requerimientos.

Por su parte, Krogstie et al. (2013) modelan la calidad basándose en la teoría semiótica. Para ello, consideran la calidad *sintáctica* como la ausencia de errores en los lenguajes utilizados durante la definición de los requerimientos, la calidad *semántica* para referirse a la integridad y

correctitud de los requerimientos, y la calidad *pragmática* relacionada al grado de comprensión de la especificación de los requerimientos por parte de los interesados.

No obstante, otras investigaciones advierten que la noción de calidad en la IR está asociada a la satisfacción de ciertas características de calidad propias de la Especificación de Requerimientos de Software (ERS), las cuales permiten controlar si el documento se elabora de forma correcta. En este sentido, las características de calidad consideradas durante la especificación de requerimientos han sido descritas en numerosos modelos de calidad (Saavedra et al., 2013). La idea principal de los modelos de calidad es desglosar el complejo concepto de calidad en características de calidad comprensibles y manejables para su evaluación. El agrupamiento de las características de calidad en un modelo permite reflexionar sobre los aspectos de calidad que se deben considerar y, además, derivar diferentes formas de medir y evaluar dichas características, a fin de conducir a una evaluación más exhaustiva de este concepto (Wagner, 2013).

Gran parte de estos modelos, fueron revisados por Saavedra et al. (2013). En su investigación, se manifiesta la existencia de una gran variedad de modelos de calidad. Algunos de ellos proponen un conjunto de características de calidad deseables para la ERS que deberían ser cubiertas en tal documento (Davis et al., 1993; Loucopoulos y Karakostas, 1995; Wilson et al., 1997; IEEE 830, 1998; Fabbrini et al., 2001a; Pohl, 2010; Swathi y Prasad, 2011; Wiegers y Beatty, 2013; Génova et al., 2013; ISO IEC IEEE 29148, 2018). Mientras que, otras propuestas sugieren una taxonomía de defectos, donde la presencia de un defecto en los requerimientos refleja la ausencia de una característica de calidad (Lanubile et al., 1998; Schneider et al., 1992).

Diversos autores reconocen a la calidad de los requerimientos como un factor determinante de éxito para un proyecto de software (Fabbrini et al., 2001b; Lami et al., 2004; Kamata y Tamai, 2007; Knauss et al., 2009). De hecho, el documento más crítico de un proyecto de software es la ERS dado que contiene los requerimientos funcionales y no funcionales del sistema a desarrollar. Además, puesto que, sirve de base para las fases posteriores de desarrollo y tiene gran impacto sobre características críticas del proyecto (tales como costos y tiempo) es que la calidad de los requerimientos incluidos en el documento debe ser analizada en detalle.

Al analizar la calidad de los requerimientos de software se deben considerar dos niveles de abstracción que incluyen: a) *Características de calidad para requerimientos individuales* y, b) *Características de calidad para conjuntos de requerimientos*. Ambas abstracciones señaladas en el estándar ISO IEC IEEE 29148 (2018), conducen a la obtención de requerimientos de software de calidad. Particularmente, este trabajo de Tesis aborda el estudio de las características de calidad para requerimientos individuales. Esto se fundamenta al considerar que la calidad de un conjunto de requerimientos que dará lugar a la generación del documento de requerimientos de software de un sistema en particular, está determinada a -priori- por la calidad de cada requerimiento individual.

2.2/ Estándar ISO IEC IEEE 29148:2018 Systems and Software Engineering - Life cycle processes - Requirements engineering

La ISO IEC IEEE 29148 (2018) es el estándar vigente aplicable al proceso de IR y, como tal, ha sustituido al ampliamente aceptado IEEE 830 (1998). El objetivo principal de este estándar es proporcionar un enfoque unificado para facilitar la gestión de los procesos de la IR durante el ciclo de vida del software. Los procesos de la IR contemplados en este estándar se resumen a continuación:

- *Proceso de Análisis de Negocio*. Este proceso consiste en definir el problema u oportunidad de negocio, caracterizar el espacio de soluciones y determinar soluciones potenciales.
- *Proceso de Definición de las Necesidades y Requerimientos de las Partes Interesadas*. Este proceso consiste en identificar las partes interesadas en el sistema a lo largo de su ciclo de vida. En este proceso también se identifican, analizan y transforman las necesidades en un conjunto común de requerimientos para expresar la interacción prevista del sistema con su entorno operativo. Además, deben definirse los requerimientos de las partes interesadas para que el sistema pueda proveer las capacidades necesarias en un entorno definido.
- *Proceso de Definición de los Requerimientos del Sistema*. Este proceso consiste en transformar las necesidades de las partes interesadas desde una perspectiva orientada al usuario en una solución técnica que satisfaga las necesidades operativas deseadas. Permite definir requerimientos que especifican atributos, funcionalidades y restricciones del sistema.

Luego, el estándar también proporciona conceptos relevantes para la IR así como a) recomendaciones para la construcción textual de los requerimientos de software, b) atributos para ellos (identificador, versión, propietario, prioridad, riesgo, fundamentación, tipo y complejidad) y c) características de calidad para los requerimientos. Además incluye, d) discusiones acerca de la aplicación iterativa y recursiva de los procesos de la IR en el ciclo de vida del software, e) proporciona una guía para la aplicación y la gestión de los procesos de la IR en las actividades que abordan requerimientos de software, definidas en las normas IEEE 15288 (2014a) e IEEE 15288 (2014b), por último, f) especifica los ítems de información resultantes de la implementación de los procesos de la IR. Una breve descripción de los ítems de información que derivan de los procesos de la IR se provee a continuación:

- *Especificación de Requerimientos de Negocio*, describe la motivación de la organización para la cual se desarrolla el sistema, define los procesos y las políticas bajo las cuales se

debe utilizar el sistema e incluye los requerimientos de alto nivel desde la perspectiva de las partes interesadas. La especificación de requerimientos de negocio, además, describe cómo la organización persigue nuevos negocios o cambia los actuales para adaptarse a un nuevo entorno empresarial.

- *Especificación de Requerimientos de las Partes Interesadas*, en base al contexto descrito en la especificación de requerimientos de negocio, este ítem de información describe cómo la organización utilizará el sistema como medio para contribuir al negocio.

- *Especificación de Requerimientos del Sistema*, identifica los requerimientos técnicos del sistema de interés. Define los requerimientos del sistema de alto nivel desde la perspectiva del dominio a partir de la información relacionada a los objetivos generales del sistema y su entorno de aplicación. Además, incluye la declaración de los requerimientos no funcionales y las limitaciones. También incluye modelos conceptuales diseñados para ilustrar el contexto del sistema, los escenarios de uso, las principales entidades del dominio y los flujos de trabajo. El propósito de este ítem de información es proporcionar una descripción de lo que el sistema debe hacer, en términos de interacciones con su entorno externo.

- *Especificación de Requerimientos del Software*, define todas las funcionalidades requeridas para el producto de software. Además, documenta las condiciones y restricciones en las que el software debe funcionar, y los enfoques de verificación previstos para los requerimientos.

A los fines de esta investigación, el estándar ISO IEC IEEE 29148 (2018) provee las bases para el análisis y evaluación de las características de calidad de los requerimientos de software. La especificación de requerimientos de software es el resultado de la transformación formal de una o más necesidades en una obligación acordada entre las partes interesadas para que una entidad desempeñe una función o posea algún atributo de calidad, dentro de los límites previamente definidos. No obstante, es esencial destacar que un requerimiento de software es más que una simple declaración bien formada y redactada sucintamente en un formato estándar o siguiendo un patrón específico. La declaración de requerimientos incluye características de calidad asociadas a estos últimos, que contribuyen al desarrollo y la gestión del proyecto de software. Para tal fin, el estándar ISO IEC IEEE 29148 (2018) formaliza un modelo de calidad para los requerimientos de software. En primer lugar, se distinguen las características de calidad para requerimientos individuales, estas son: *Necesario*, *Apropiado*, *No ambiguo*, *Completo*, *Singular*, *Factible*, *Verificable*, *Correcto* y *Conforme*. En segundo lugar, se proveen características de calidad para un conjunto de requerimientos que incluye: *Completo*, *Consistente*, *Alcanzable*, *Comprensible* y *Verificable*. Tal como se menciona en secciones anteriores, esta investigación abordará las características de calidad para requerimientos individuales.

2.3/ Características de Calidad para Requerimientos Individuales

En esta sección se describen y analizan las características de calidad propuestas por el estándar ISO IEC IEEE 29148 (2018) para requerimientos individuales. Cada característica analizada incluye descripciones de requerimientos expresados en lenguaje natural, que permiten ilustrar el logro de la misma durante el proceso de especificación.

Cabe mencionar que los requerimientos utilizados para ejemplificar las características de calidad pertenecen al dominio de los sistemas bancarios y están escritos en inglés. Esto es así, dado que es el idioma ampliamente elegido en la industria de software para homogeneizar la comunicación entre las partes interesadas y para documentar y gestionar proyectos. Esto surge como consecuencia de la tendencia constante e irreversible de las últimas décadas, hacia la globalización de los negocios, particularmente, de las empresas dedicadas al desarrollo de productos de software. Este fenómeno, ha generado escenarios de desarrollo global, donde las partes interesadas se encuentran geográficamente distribuidas, con zonas horarias e idiomas diversos (Herbsleb y Moitra, 2001). A continuación, se proporciona una descripción detallada de cada una de las características de calidad abordadas en esta investigación.

2.3.1/ Característica de Calidad Necesario

Un requerimiento de software cumple con la *característica de calidad necesario* si define una funcionalidad, característica, restricción o factor de calidad esencial de forma tal que, si no es incluida en la ERS, existirá una deficiencia en el sistema a desarrollar, que no podrá ser satisfecha mediante la implementación de otros requerimientos.

Esta característica de calidad sólo puede garantizarse revisando cada requerimiento en relación a la misión, las metas, los objetivos y las restricciones definidas para la entidad que da lugar a tal requerimiento. Si el requerimiento no se relaciona con una o más necesidades de la entidad entonces, el requerimiento es innecesario. Por lo tanto, no resulta fácil determinar si un requerimiento cumple dicha característica de calidad sin una noción clara del contexto, los requerimientos asociados y el dominio de implementación. Por otro lado, la inclusión de requerimientos innecesarios puede conducir al retrabajo, costos y riesgos adicionales que afectan negativamente el proyecto de desarrollo de software. La Figura 2.1 presenta la especificación de los requerimientos ***Requerimiento 2.1*** y ***Requerimiento 2.2*** que permiten ejemplificar la *característica de calidad necesario*.

El ***Requerimiento 2.1*** establece que el sistema debe generar y verificar un certificado SSL válido para las transferencias bancarias a través del protocolo HTTPS. Esta característica de seguridad es importante en el contexto de las transacciones bancarias en línea, ya que asegura que la información transmitida esté protegida contra posibles intrusiones o manipulaciones por parte de terceros no autorizados. Además, el uso del protocolo HTTPS en la URL de todas las

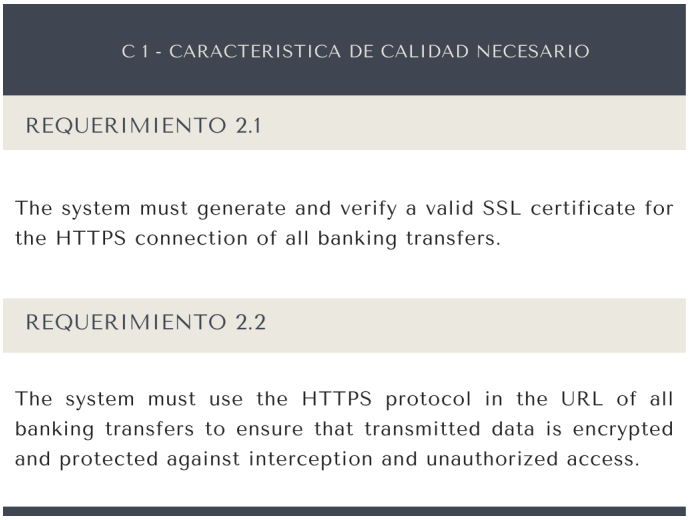


Figura 2.1: Característica de calidad Necesario.

transferencias bancarias descrito en el ***Requerimiento 2.2*** garantiza que los datos se transmitan de forma cifrada y segura, protegiéndolos de la intercepción y el acceso no autorizado. La relación entre ambos requerimientos señala que la inclusión de ***Requerimiento 2.1*** es necesaria dado que sin su consecución no es posible alcanzar ***Requerimiento 2.2***. De otra forma, si el ***Requerimiento 2.1*** fuera eliminado de la ERS, los requerimientos restantes no serían suficientes para satisfacer las necesidades de la entidad que ha generado el ***Requerimiento 2.2***.

2.3.2/ Característica de Calidad Apropiado

Un requerimiento de software cumple con la *característica de calidad apropiado* si éste se expresa con un nivel de detalle adecuado y consistente, sin especificar cómo debe implementarse. Esto se refiere a que la intención debe ser específica y el detalle del requerimiento debe ajustarse al nivel de abstracción en el que se establece. Esto último también incluye evitar restricciones innecesarias respecto al diseño arquitectónico para facilitar la independencia de implementación. La definición de un requerimiento apropiado debe enfocarse en el qué y no el cómo, es decir, capturar la necesidad que se desea resolver, sin incluir soluciones.

La Figura 2.2 presenta el ***Requerimiento 2.3*** el cual especifica que el sistema debe permitir a los usuarios transferir fondos entre cuentas bancarias propias y de terceros, indicando el monto y la cuenta de destino. Además, señala que la interfaz de usuario debe ser clara y fácil de usar, con instrucciones detalladas y retroalimentación en tiempo real para confirmar la transacción. El requerimiento no describe cómo debe implementarse la solución, sólo indica el resultado esperado, dejando libertad al equipo de desarrollo para elegir la solución óptima. Por lo tanto, se considera que ***Requerimiento 2.3*** cumple con la *característica de calidad apropiado*, dado que no incluye soluciones en su definición.

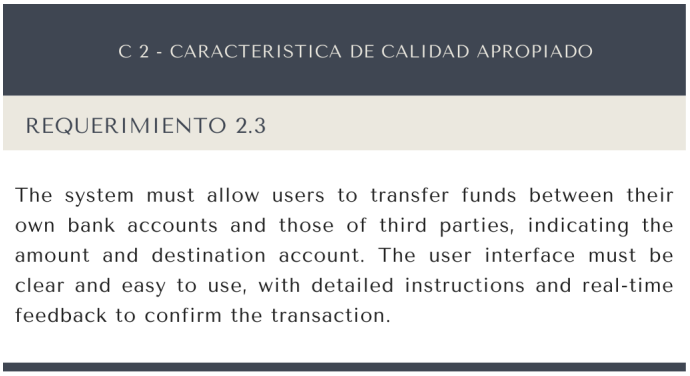


Figura 2.2: Característica de calidad Apropiado.

2.3.3/ Característica de Calidad No Ambiguo

Un requerimiento de software cumple con la *característica de calidad no ambiguo* si está enunciado de forma tal que sólo permite una única interpretación válida. Por lo que, el requerimiento debe ser fácil y simple de entender. La intención del requerimiento debe ser comprendida de la misma manera por todos los involucrados en el proyecto. En contraste a lo mencionado, los requerimientos ambiguos dan lugar a múltiples interpretaciones no previstas que generan malentendidos y, en consecuencia, esto puede conducir a que las expectativas y necesidades del cliente no sean satisfechas.

En general, si un requerimiento es verificable, es decir, si es posible establecer un caso de prueba para tal requerimiento sin hacer suposiciones acerca del significado o interpretación de su definición, entonces éste no es ambiguo. No obstante, debido a la flexibilidad que caracteriza al lenguaje natural, resulta difícil eliminar por completo la ambigüedad de los requerimientos. En tal caso, es posible añadir en la documentación los fundamentos del requerimiento para proporcionar una visión adicional de la intención y así, reducir la ambigüedad. Por otro lado, también es posible mitigarla evitando el uso de conjunciones, cláusulas abiertas, infinitivos superfluos y acrónimos, entre otros, utilizados en la descripción de los requerimientos que fomentan la participación activa del receptor dando lugar, a que este genere su propia interpretación.

Las cláusulas pueden conducir a la definición de requerimientos ambiguos y no verificables que están abiertos a interpretación y que no reflejan con precisión las expectativas de las partes interesadas. Además desde un punto de vista contractual, los requerimientos con estas frases podrían interpretarse como opcionales.

La Figura 2.3 presenta el ***Requerimiento 2.4*** el cual establece que el sistema debe permitir al usuario ver el saldo disponible de cada una de sus cuentas bancarias en una sola interfaz. Además, se establece que el usuario debe tener la posibilidad de filtrar y buscar transacciones específicas por fecha, tipo e importe, lo que significa que el sistema debe proporcionar las herramientas necesarias para realizar estas acciones de manera efectiva. El requerimiento no es ambiguo porque especifica claramente lo que el sistema debe hacer y proporciona detalles específicos sobre la funcionalidad que se espera.

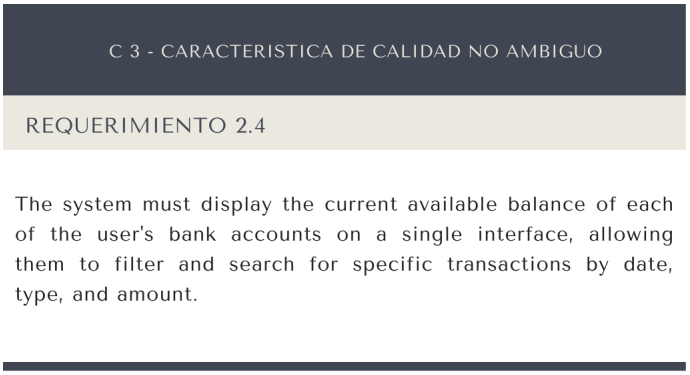


Figura 2.3: Característica de calidad No Ambiguo.

2.3.4/ Característica de Calidad Completo

Un requerimiento de software cumple con la *característica de calidad completo* si describe una funcionalidad, característica, restricción o factor de calidad de forma suficiente. Es decir, no se requiere otra información para su comprensión. La declaración de un requerimiento completo debe ser una frase que no precise referencias a otras declaraciones para ser entendida. Aunque a pesar de esto, el requerimiento puede hacer referencia a otros documentos, tales como normas y reglamentos.

De acuerdo con el estándar ISO IEC IEEE 29148 (2018) para alcanzar esta característica de calidad es recomendable evitar el uso de expresiones del tipo TBD (To Be Defined), TBS (To Be Specified) o TBR (To Be Resolved) en el proceso de especificación de requerimientos para indicar trabajo en curso. En caso de no ser posible, debe quedar claro quiénes son los responsables de los aspectos sin resolver y cuándo serán resueltos.

Cabe agregar, que determinar si un requerimiento cumple con la *característica de calidad completo*, puede requerir una comprensión exhaustiva del contexto y dominio de implementación. Por último, aplicar patrones o guías de estilo puede facilitar la obtención de requerimientos completos, dado que su uso proporciona sugerencias sobre la estructura gramatical y los aspectos que debería incluir la declaración para cumplir con tal propiedad de calidad. La Figura 2.4 presenta la definición del ***Requerimiento 2.5*** para ilustrar la *característica de calidad completo*.

El ***Requerimiento 2.5*** establece los criterios necesarios para garantizar el acceso seguro a la cuenta de usuario de un sistema bancario en línea. Se especifican de forma clara las condiciones para creación de contraseña y restricciones sobre los datos personales del usuario que no deben ser incluidos en las credenciales de acceso. Por lo tanto, el ***Requerimiento 2.5*** cumple con la *característica de calidad completo*.

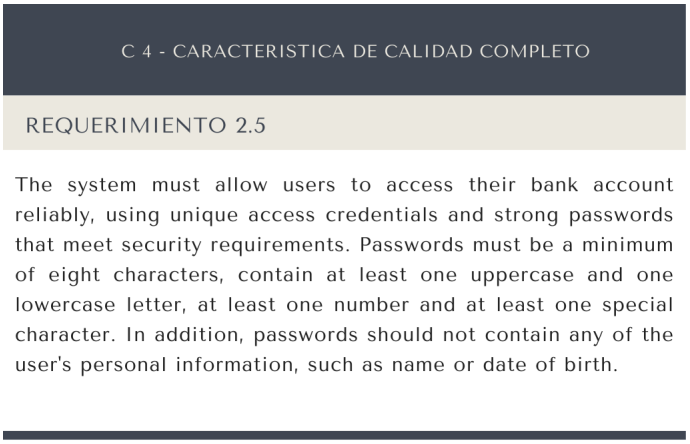


Figura 2.4: Característica de calidad Completo.

2.3.5/ Característica de Calidad Singular

Un requerimiento de software cumple con la *característica de calidad singular* si expresa una única funcionalidad, característica, restricción o factor de calidad. No obstante, también puede incluir múltiples condiciones bajo las cuales el requerimiento debe cumplirse. En contraste, un requerimiento no es singular si describe múltiples hechos aislados o débilmente acoplados que pueden fácilmente ser divididos en diversos requerimientos individuales.

La Figura 2.5 presenta la definición de ***Requerimiento 2.6*** el cual cumple con la *característica de calidad singular*, ya que se enfoca en un único objetivo o funcionalidad específica del sistema, que es permitir al usuario solicitar una tarjeta de crédito adicional a través de la plataforma de un sistema bancario en línea y especifica claramente los datos que el usuario debe proporcionar para realizar dicha solicitud.

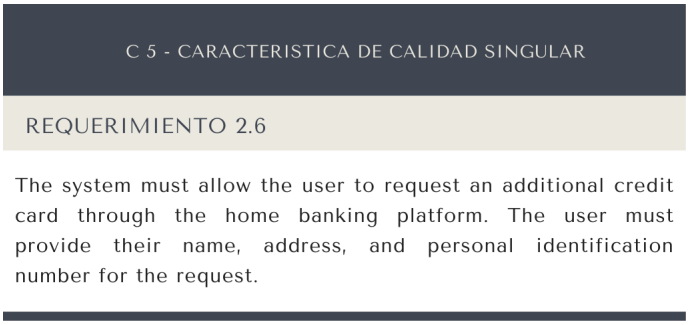


Figura 2.5: Característica de calidad Singular.

2.3.6/ Característica de Calidad Factible

Un requerimiento de software cumple con la *característica de calidad factible* si puede desarrollarse conforme a las capacidades y limitaciones del proyecto o sistema en cuestión. Se consideran aspectos tales como costos, calendario, aspectos técnicos, normativa legal, ética y

entorno operativo, con un riesgo aceptable. Por lo general, es difícil determinar si un requerimiento es factible sin un análisis previo de las posibles soluciones. Es por ello, que a menudo la factibilidad se evalúa en términos de riesgo aceptable. Una buena herramienta para evaluar el riesgo es la creación de prototipos, simulaciones, evaluaciones empíricas, entre otras, esto contribuye a determinar la viabilidad de los requerimientos de software. La Figura 2.6 presenta el ***Requerimiento 2.7*** para ejemplificar la *característica de calidad factible*.

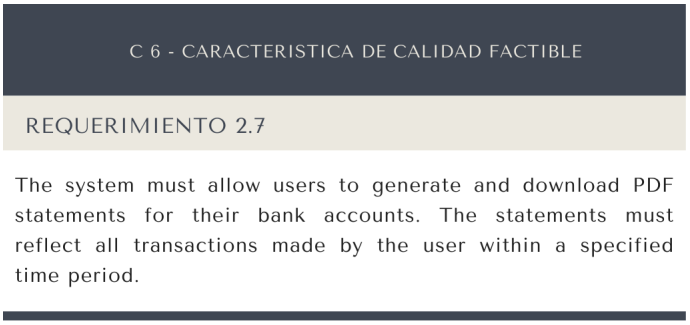


Figura 2.6: Característica de calidad Factible.

El ***Requerimiento 2.7*** especifica la necesidad de generar y descargar extractos bancarios en formato PDF de una cuenta particular, una función común en los sistemas bancarios en línea y factible de implementar con tecnologías actuales. Además, se solicita que los extractos reflejen todas las transacciones realizadas por el usuario en un periodo de tiempo determinado. En este sentido, la verificación de este tipo de condiciones, que incluyen un rango definido de valores con límites superiores e inferiores, como puede ser el período de fecha *(Fecha desde - Fecha Hasta)* considerado para la generación del extracto también resulta factible. Por lo tanto, se considera que el ***Requerimiento 2.7*** cumple con la característica de calidad factible.

2.3.7/ Característica de Calidad Verificable

Un requerimiento de software cumple con la *característica de calidad verificable* si está estructurado y redactado de tal manera, que los involucrados pueden comprobar que el sistema, funcionalidad o característica desarrollada efectúa aquello que se ha documentado. Algunos de los métodos de verificación estándar utilizados son la inspección, análisis y pruebas. Adicionalmente, Pohl (2010) señala que la inclusión de criterios de aceptación conduce a la obtención de requerimientos verificables. Las causas más comunes que conllevan a la obtención de requerimientos no verificables son:

- Ausencia de detalles respecto al comportamiento funcional, condiciones y estados correctos;
- Falta de precisión en los rangos de rendimiento aceptables;
- Uso de términos ambiguos.

Por último, cabe mencionar que aquellos requerimientos que no cumplen con la *característica de calidad factible* difícilmente sean verificables, puesto que, no se puede verificar aquello que no es posible desarrollar o realizar. La Figura 2.7 presenta el ***Requerimiento 2.8*** para ejemplificar la *característica de calidad verificable*.

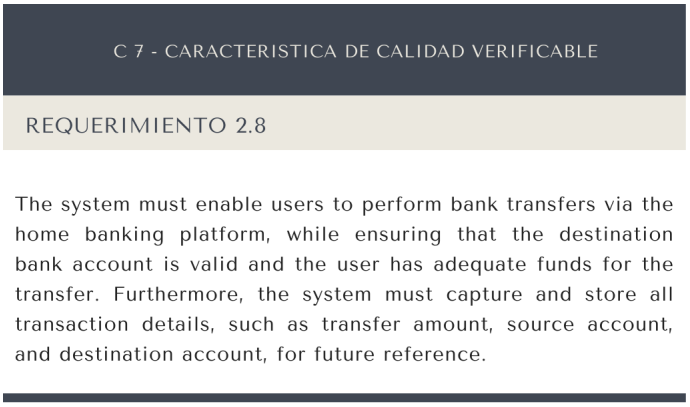


Figura 2.7: Característica de calidad Verificable.

El ***Requerimiento 2.8*** se considera verificable dado que es posible establecer casos de prueba para este sin hacer suposiciones acerca del significado o interpretación de su definición. Particularmente, es posible validar la cuenta bancaria de destino con la información de la cuenta bancaria ingresada por el usuario, como el número de cuenta y el código de identificación bancaria. Por su parte, se puede verificar si un usuario tiene fondos suficientes para realizar una transferencia mediante el uso de tecnologías de intercambio de datos seguras con el banco. También es posible verificar si el sistema registra y almacena de manera precisa y completa la información de la transacción, incluyendo el monto transferido, la cuenta de origen y la cuenta de destino a través consultas a la base de datos.

2.3.8/ Característica de Calidad Correcto

Un requerimiento de software cumple con la *característica de calidad correcto* si representa con exactitud la característica, funcionalidad, restricción o factor de calidad que se precisa. Un requerimiento correcto debe demostrar que tal y como está redactado, dará lugar a la satisfacción de la necesidad de aquella entidad que le dio origen. La Figura 2.8 presenta el ***Requerimiento 2.9*** para ilustrar la *característica de calidad correcto*.

El ***Requerimiento 2.9*** para el proceso de cambio de correo electrónico en el sistema bancario está articulado con un enfoque en claridad, integridad y seguridad. Comienza especificando una acción del usuario (cambiar el correo electrónico), luego avanza para garantizar la unicidad del nuevo correo, asegura su validación a través de una confirmación y culmina en la actualización adecuada de la base de datos del banco. La claridad se demuestra ya que el requerimiento describe explícitamente el proceso paso a paso sin ambigüedades. La integridad se refleja en cómo abarca cada aspecto esencial, desde la iniciación hasta la actualización de la

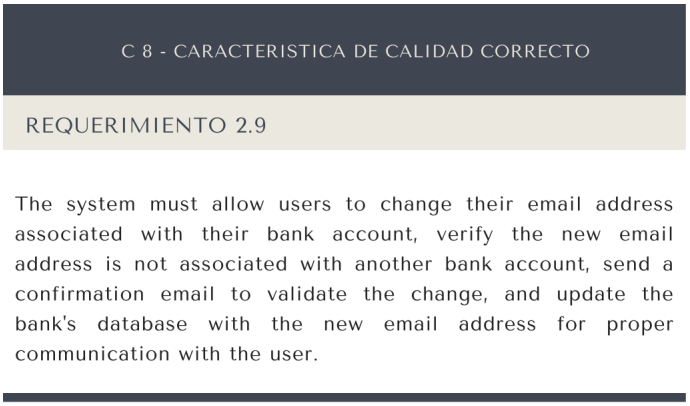


Figura 2.8: Característica de calidad Correcto.

base de datos. La seguridad está implícitamente enfatizada al garantizar la unicidad del correo electrónico, lo que protege contra potenciales duplicidades de cuentas o fraudes. Al garantizar que el proceso de cambio de correo electrónico sea completo y claro, aborda posibles vulnerabilidades del sistema y promueve la confianza y satisfacción del usuario. Esta alineación con las necesidades del usuario y la integridad del sistema justifica su correctitud en el contexto de requerimientos de calidad.

2.3.9/ Característica de Calidad Conforme

Un requerimiento de software cumple con la *característica de calidad conforme* si la definición del mismo se ajusta a un estilo de escritura, formato y/o plantilla previamente acordada para su documentación.

En general, el uso de patrones para estructurar requerimientos, tal como muestra la Figura 2.9, facilita la obtención de la *característica de calidad conforme*. Cuando los requerimientos en una organización tienen el mismo formato y estructura resulta más fácil su revisión y comprensión. En relación con esto, el estándar ISO IEC IEEE 29148 (2018) recomienda que cada organización disponga de una guía o esquema para abordar cuestiones como la selección y definición de los patrones que deben utilizarse para escribir requerimientos, la selección y el uso de atributos, las abreviaturas y acrónimos, la disposición y el uso de figuras y tablas, así como también para la numeración de los requerimientos. En la Figura 2.10 se presenta el ***Requerimiento 2.10*** para ilustrar la *característica de calidad conforme*.

La definición de ***Requerimiento 2.10*** denota el uso de una guía de estilo apropiada, lo que facilita su comprensión y verificación. Este requerimiento cumple con la *característica de calidad conforme* porque especifica claramente cómo el sistema debe comportarse cuando un usuario ingresa una contraseña incorrecta y cómo debe registrar y proteger la seguridad de la cuenta después de varios intentos fallidos. Además, es medible y verificable, ya que se pueden establecer pruebas para verificar que el sistema muestra un mensaje de error cuando se ingresa una contraseña incorrecta y bloquea temporalmente la cuenta después de tres intentos fallidos.

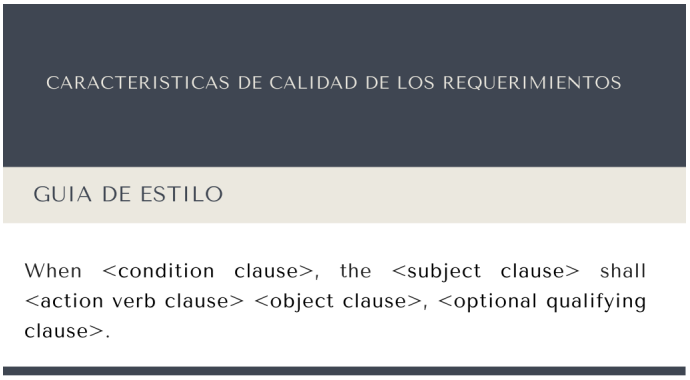


Figura 2.9: Guía de estilo para especificar requerimientos.

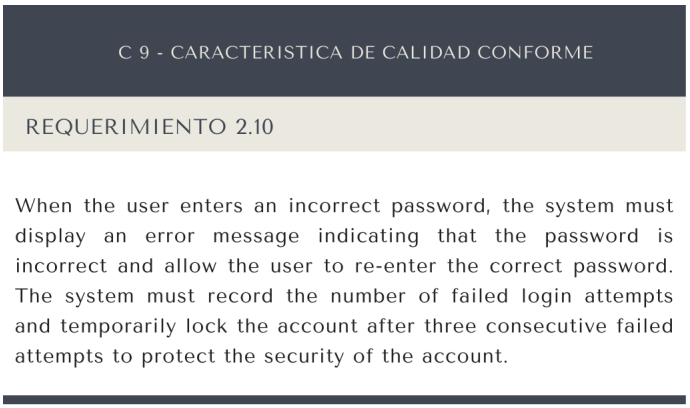


Figura 2.10: Característica de calidad Conforme.

2.4/ Especificación de Requerimientos de Software en Lenguaje Natural

La obtención de los requerimientos es un proceso complejo e iterativo en el que los ingenieros deben responder al doble desafío de descubrir y formalizar las necesidades de los interesados (Génova et al., 2013). Los desafíos asociados al descubrimiento de los requerimientos incluyen (Pressman y Maxim, 2016):

- *Problemas de alcance*, vinculados a una mala definición del límite o alcance del sistema, especificación de detalles técnicos innecesarios por parte de los interesados que pueden causar confusión sobre los objetivos generales del sistema.
- *Problemas de entendimiento*, asociados a las dificultades de las partes interesadas para expresar las necesidades, solicitud de requerimientos ambiguos, comprensión parcial del dominio del problema y omisión de la información.
- *Problemas de volatilidad*, a causa de requerimientos inconsistentes.

El éxito de este proceso conlleva una intensa colaboración y comunicación con los interesa-

dos durante todo el ciclo de vida del software. Especialmente, durante las fases tempranas del proyecto, donde se expresan y definen los requerimientos de software. La necesidad de reducir las brechas de comunicación entre los grupos de interesados ha posicionado al lenguaje natural como la principal técnica para expresar los requerimientos de software. En consecuencia, su uso favorece la comunicación y colaboración entre las partes interesadas, dado que no requiere conocimientos técnicos previos y no posee limitaciones respecto a los conceptos que se pueden expresar.

El lenguaje natural es ampliamente utilizado en la IR para obtener y documentar los requerimientos de un proyecto de software (Fabbrini et al., 2001b). Denger et al. (2003), señalan que su uso es predominante respecto a otros tipos de técnicas formales, tales como tablas, diagramas, notaciones formales, pseudocódigo, entre otras. Por su parte, Mich et al. (2004) reportan como resultado de una encuesta realizada a nivel global en la industria del software, que el 95% de los participantes afirma representar sus requerimientos utilizando lenguaje natural. De ellos, sólo el 16% utiliza lenguaje natural estructurado, tales como plantillas y formularios, mientras que el 79% restante aplica lenguaje natural.

El lenguaje natural presenta tres características importantes que promueven su uso en la industria de software (Kamsties y Peach, 2000):

- *Flexible*. La flexibilidad del lenguaje natural permite a los involucrados transmitir los requerimientos sin restricciones y con diferentes niveles de detalle, que varían desde lo más abstracto a lo más específico. Esta característica, además, facilita el refinamiento de los requerimientos durante las actividades de obtención y especificación.

- *Comprensible*. El entendimiento de los requerimientos documentados en lenguaje natural es accesible a las partes interesadas sin ningún entrenamiento previo o uso de herramientas especiales.

- *Universal*. Su uso contribuye a la comunicación y descripción de los requerimientos de las partes interesadas en cualquier dominio de aplicación.

Otros motivos que promueven su uso se relacionan con las características críticas de tiempo y presupuesto que demanda la competencia global para obtener productos de software. El uso del lenguaje natural bajo estas restricciones permite centrar los esfuerzos en la definición del problema y la obtención de los requerimientos involucrados en la solución, por encima del uso de técnicas formales y/o de modelado (Nikora y Balcom, 2009). Consecuentemente, por estas razones, el lenguaje natural es utilizado con frecuencia en la industria del software para representar los requerimientos de forma práctica y rápida.

Desafortunadamente, el lenguaje natural también presenta algunas características negativas tales como la vaguedad y ambigüedad que desfavorecen su uso. De modo que el lenguaje natural puede dar lugar a la definición de requerimientos poco claros que afectan la calidad. En efecto los requerimientos pueden estar sujetos a múltiples interpretaciones, que generan

malentendidos y conflictos entre los interesados. Esta situación, puede conducir a la inserción de defectos, que repercuten de forma transversal y negativa en las fases posteriores de desarrollo. Es por ello, que resulta clave la integración de nuevas tecnologías a la IR para enriquecer técnicas y procedimientos tradicionales y, con ello, favorecer la calidad de los requerimientos de software.

2.5/ Métodos para Evaluar la Calidad de los Requerimientos de Software

En la literatura se pueden hallar diversos estudios que proponen evaluar la calidad de los requerimientos en lenguaje natural. Kummler (2021) señalan tres principales enfoques de investigación:

- *Sistemas basados en reglas*. Las propuestas bajo este enfoque coinciden en el uso de técnicas de procesamiento de lenguaje natural para obtener secuencias representativas de los requerimientos de software que luego, serán procesados a través de algoritmos. Estos últimos, presentan mecanismos de control representados por sentencias condicionales, que permiten detectar o restringir el uso de reglas gramaticales, expresiones regulares, corpus de palabras y/o frases y a partir de ello determinar indicadores cuantitativos de calidad (Wilson et al., 1997; Fabbrini et al., 2001a; Verma y Kass, 2008; Holtmann et al., 2011; Génova et al., 2013; Konig et al., 2017).
- *Sistemas de modelado y formalización*. Las propuestas bajo este enfoque están orientadas a la transformación de los requerimientos en lenguaje natural en modelos formales y especificaciones lógicas, mediante el análisis lingüístico y el uso de ontologías (Ilieva y Ormandjieva, 2005; Arellano et al., 2015; Bhatia et al., 2013; Ghosh et al., 2016; Kaiya y Saeki, 2006; Huertas y Juárez-Ramírez, 2013).
- *Sistemas de clasificación automática*. Las propuestas bajo este enfoque utilizan técnicas de aprendizaje automático con el fin de clasificar los requerimientos y así determinar cuantitativamente su calidad (Ormandjieva et al., 2007; Yang et al., 2012; Parra et al., 2015). Para ello, aplican previamente métodos de procesamiento de lenguaje natural (lemmatization, tokenization, stemming) para representar los requerimientos en un formato de entrada adecuado para los algoritmos de aprendizaje automático.

La comunidad de la IR ha propuesto múltiples iniciativas para modelar, evaluar y medir la calidad de los requerimientos expresados en lenguaje natural. De hecho, en la literatura hay una amplia variedad de enfoques, todos ellos orientados en una dirección similar: determinar las características de calidad relevantes y definir métricas para su evaluación.

A pesar de que las estrategias basadas en reglas pueden ser útiles, en la evaluación de la calidad de los requerimientos a menudo resultan insuficientes o poco flexibles. Además, estas

estrategias son señaladas por ser arbitrarias en la definición de los parámetros utilizados para las mediciones, así como rígidas en la combinación de métricas para evaluar las diferentes características de calidad (Moreno et al., 2020). Esto es así, dado que la evaluación de la calidad en este tipo de enfoques de investigación está circunscrita sólo a la detección de palabras clave, corpus de palabras y reglas gramaticales previamente definidas, cuya presencia en un requerimiento se asocia a un valor de penalización que luego se usa para calcular un conjunto de métricas y con esto determinar la calidad. Estos mecanismos de control no contemplan posibles excepciones ni consideraciones semánticas y/o sintácticas sobre las que un ingeniero de software puede reflexionar al evaluar un requerimiento en un determinado dominio, proyecto u organización. De hecho, algunas de las estrategias utilizadas para abordar la calidad de los requerimientos pueden considerarse obsoletas o de rendimiento cuestionable respecto a las exigencias y características que surgen del contexto actual (tiempo, complejidad, volumen de datos, criticidad).

Afortunadamente, los avances en la Inteligencia Artificial (IA) y la disponibilidad de grandes volúmenes de datos han reducido significativamente las barreras de entrada al uso del procesamiento del lenguaje natural. Esto crea oportunidades sin precedentes para aplicar dichas técnicas a la práctica de la IR con el objetivo de ayudar a analizar automáticamente los documentos de requerimientos. En este sentido, se realizó un análisis preliminar respecto al uso de las técnicas de la IA en el área de IR, a fin de establecer el alcance y los objetivos para este trabajo de Tesis.

2.6/ Integración de la Inteligencia Artificial en la Ingeniería de Requerimientos

En los últimos años, la IA ha experimentado un crecimiento explosivo, principalmente, a causa del renovado interés sobre su aplicación en diversos dominios. Muchos de estos avances están asociados a la disponibilidad de grandes conjuntos de datos, la capacidad de entrenar y aprender a gran escala usando GPUs, y el surgimiento de redes neuronales más grandes y complejas, lo que ha permitido obtener resultados de mayor precisión (Feldt et al., 2018). Esta situación ha motivado a diversos autores a explorar sobre su uso en el proceso de desarrollo de software para optimizar y automatizar tareas que demandan esfuerzos intensivos.

Estudios recientes han demostrado resultados prometedores al combinar técnicas tradicionales con aquellas que derivan de la IA para brindar soluciones a las tareas involucradas en el desarrollo de software. En este sentido, la IR se ha visto ampliamente beneficiada a partir de la implementación de dichas técnicas. Estos aportes fueron analizados a través de un mapeo sistemático de la literatura (Gramajo et al., 2020a). Dicho estudio ha permitido determinar el estado del arte respecto a la aplicación de las técnicas de aprendizaje automático supervisado en la IR. Las contribuciones de esta investigación se mencionan a continuación:

- Revisión comprensiva de las investigaciones que aplican técnicas de aprendizaje supervisado en la IR;
- Identificación y análisis de las técnicas de aprendizaje supervisado utilizadas en las investigaciones para abordar las actividades de la IR;
- Identificación de las fuentes de datos utilizadas en las propuestas de investigación y, por último;
- Detección de oportunidades y lagunas de investigación en el campo del aprendizaje supervisado aplicado a la IR.

2.6.1/ Revisión Comprensiva de Estudios Seleccionados

Los artículos identificados demuestran que el aprendizaje supervisado es utilizado en la actualidad por ingenieros e investigadores para obtener nuevas perspectivas y soluciones a problemas que anteriormente se encontraban fuera de su alcance.

En esta investigación se identificaron métodos y propuestas que apoyan las tareas de obtención y especificación de requerimientos. La obtención de requerimientos es la tarea que determina las necesidades y restricciones del producto de software que se va a desarrollar, a través de la interacción con los clientes, usuarios y diversas fuentes de información. Estas fuentes de información pueden incluir grupos de interés, documentos tanto dentro como fuera del universo de información y software interno o externo. La recopilación de los requerimientos se lleva a cabo mediante entrevistas, cuestionarios y otras técnicas. Es una tarea compleja, porque requiere una comunicación eficiente, a fin de especificar posteriormente lo recopilado en un documento sistematizado de requerimientos. Es por ello que los requerimientos deben concebirse sin ambigüedades para definir de manera correcta el comportamiento del sistema esperado por las partes interesadas. En esta revisión se identificaron cuatro (4) propuestas que abordan los desafíos relacionados con la detección de incertidumbre, frases especulativas (Yang et al., 2012; Knauss et al., 2016) y ambigüedad (Yang et al., 2010a, 2011) en documentos de requerimientos y artefactos escritos en lenguaje natural. Estas propuestas tienen por objeto ayudar a los ingenieros a identificar los requerimientos propensos a generar malentendidos entre las partes interesadas.

Por otro lado, determinar la consistencia acerca de un documento de requerimientos también es una tarea abordada mediante las técnicas de aprendizaje supervisado (Nikora y Balcom, 2009; Ott, 2013). Sólo se identificaron dos (2) estudios relacionados a este tema. Estas propuestas utilizan técnicas de procesamiento de lenguaje natural y aprendizaje supervisado para identificar tipos específicos de requerimientos dentro de un conjunto de especificaciones y generar subconjuntos. Esto facilita el análisis de cada subconjunto respecto a aspectos, tales como consistencia, integridad y ambigüedad. La agrupación de requerimientos en muestras de menor tamaño minimiza la probabilidad de errores durante su análisis.

Otras investigaciones abordan la clasificación automática de los requerimientos funcionales y las diversas subcategorías de los requerimientos no funcionales. Particularmente, esto se debe al hecho de que las partes interesadas -al igual que los ingenieros de requerimientos- usan diversas terminologías y construcciones sintácticas y semánticas para describir el mismo tipo de requerimiento. Dados estos desafíos, es imperativo desarrollar nuevas técnicas y herramientas que puedan respaldar la clasificación de requerimientos. En este estudio preliminar se identificaron siete (7) propuestas que apuestan a resolver esta tarea mediante la aplicación de técnicas de aprendizaje supervisado (Slankas y Williams, 2013a,b; Jindal et al., 2016; Li, 2017; Kurtanović y Maalej, 2017; Dekhtyar y Fong, 2017; Abad et al., 2017).

Por otro lado, Merten et al. (2016) en su propuesta aplican técnicas de aprendizaje supervisado y procesamiento de lenguaje natural para clasificar los requerimientos de software presentes en herramientas de gestión de proyectos y seguimiento de errores e incidencias.

Los documentos de requerimientos, además de contener especificaciones, poseen contenido auxiliar, tales como resúmenes, explicaciones, referencias a otros documentos, entre otros, que permiten comprender el dominio del sistema a desarrollar. Se detectó un (1) estudio que aborda la categorización automática del contenido de la ERS para distinguir los requerimientos de software de la información auxiliar provista (Winkler y Vogelsang, 2016). En tal sentido, los autores señalan que la discriminación del contenido de la ERS puede simplificar el trabajo de los ingenieros, dado que permite identificar fácilmente los requerimientos que estarán involucrados en tareas posteriores en el proceso de desarrollo del software. Por ejemplo, en la definición de casos de prueba sin necesidad de revisar nuevamente el documento de requerimientos.

Otras investigaciones tienen como foco de estudio los enlaces de trazabilidad, fundamentales para el mantenimiento de un producto de software, porque establecen vínculos entre los artefactos de software. Esto permite que un requerimiento sea trazable desde su definición y durante todo el desarrollo del software, lo cual garantiza una adecuada gestión de cambios y evaluación de impacto en el resto del sistema. La identificación y clasificación de la trazabilidad entre los requerimientos es abordada en seis (6) propuestas (Li et al., 2017; Cleland-Huang et al., 2010; Gokyer et al., 2008; Mills y Haiduc, 2017; Sardinha et al., 2013; Atas et al., 2018).

Las reglas de negocio constituyen las políticas que rigen las operaciones en una organización. La identificación y comprensión de las reglas de negocio es una tarea importante del proceso de la IR. Sin embargo, resulta desafiante ya que estas reglas, a menudo, no se establecen explícitamente en los documentos de requerimientos. En caso de que las reglas de negocio sean explícitas, pueden ser vagas o de naturaleza no atómica. Los hallazgos en esta investigación demuestran que el aprendizaje supervisado puede aplicarse para detectar y categorizar las reglas de negocio de las especificaciones de requerimientos (Sharma et al., 2014). Se identificó sólo un (1) estudio que propone una solución a dicho desafío.

Otra de las tareas más desafiantes en el desarrollo de un proyecto de software, es la obtención de requerimientos con calidad. Los requerimientos erróneos no detectados en fases tempranas pueden provocar costos adicionales, insatisfacción de las partes interesadas y retra-

sos en la entrega del producto software. Consecuentemente, esto podría generar la cancelación del proyecto. Por estas razones, el estudio de la calidad de los requerimientos resulta clave. En este estudio preliminar se identificaron cinco (5) propuestas que abordan dicha temática de investigación (Hussain et al., 2007; del Águila y Del Sagrado, 2011; Parra et al., 2015; Hayes et al., 2015; Dargan et al., 2016).

Por otro lado, se detectó un enfoque que emplea el aprendizaje supervisado para cuantificar el impacto de los requerimientos no funcionales en la estimación del esfuerzo para el desarrollo de un proyecto de software (Abdukalykov et al., 2011). También, se identificó una propuesta que aborda el análisis de los requerimientos para extraer automáticamente información semántica (Wang, 2016). Otros aplican el aprendizaje supervisado para predecir fallas en las funcionalidades solicitadas por las partes interesadas (Fitzgerald et al., 2012). Algunas investigaciones utilizan el aprendizaje supervisado para hacer predicciones respecto a la necesidad de revisión en los documentos de requerimientos, con el objetivo de ayudar a los ingenieros a determinar si una especificación de requerimientos es lo suficientemente estable o necesita revisión adicional (del Sagrado y del Aguila, 2018).

La revisión comprensiva de los estudios seleccionados demuestra que las técnicas de aprendizaje supervisado en la IR se centran en seis grandes categorías: (a) Análisis de factibilidad de los requerimientos (ASR); (b) Identificación de reglas de negocio (BRI); (c) Clasificación de requerimientos (SRC); (d) Resolución de problemas lingüísticos que derivan del uso del lenguaje natural en documentos de requerimientos y artefactos (NLI); (e) Trazabilidad (TR) y (f) Calidad de requerimientos (QSR).

2.6.2/ Extracción y Mapeo de Datos

A partir del análisis de los estudios seleccionados, es notable el amplio abanico de aplicaciones del aprendizaje supervisado en el campo de la IR. Especialmente, se destacan las propuestas enfocadas en la clasificación de requerimientos. También se distinguen las investigaciones destinadas a resolver problemas tales como la ambigüedad y consistencia en los documentos de requerimientos y artefactos escritos en lenguaje natural. Por último, se observa la aplicación de estas técnicas en propuestas que abordan la trazabilidad de los requerimientos.

La Figura 2.11 muestra la distribución de artículos que emplean el aprendizaje supervisado en función a las categorías definidas anteriormente. Se puede apreciar que, a pesar del surgimiento y liberación de nuevas tecnologías, así como librerías en el año 2002 que permitieron explorar el uso de las técnicas de aprendizaje automático en diversos dominios, recién a partir del año 2007 comenzaron a ser aplicadas en el área de la IR. Las propuestas iniciales, estuvieron orientadas a la resolución de desafíos relacionados a la calidad (QSR) y trazabilidad de los requerimientos (TR). Sin embargo, las investigaciones sobre estos tópicos no mostraron grandes variaciones a lo largo del tiempo. Por otro lado, los estudios que abordaron la clasificación de requerimientos (SRC) denotan un mayor crecimiento en el año 2017. También cabe

destacar que la detección de problemas que derivan del uso del lenguaje natural en documentos y artefactos de requerimientos (NLI) resultó ser uno de los retos más abordados hasta el año 2013.

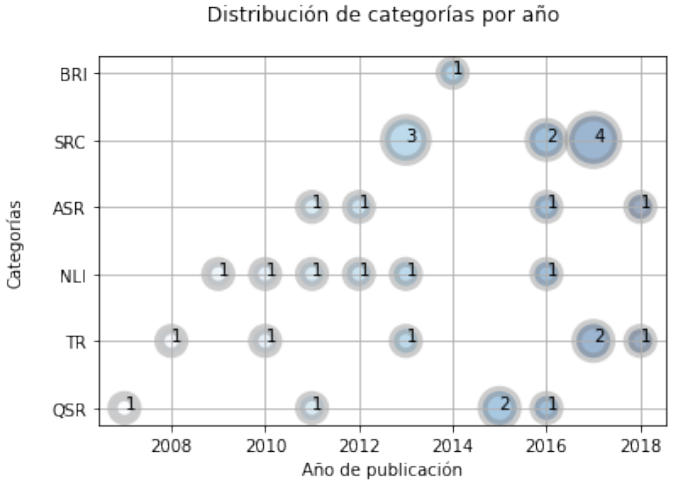


Figura 2.11: Distribución de estudios por año y categoría.

Durante el análisis de las propuestas se detectó la aplicación de una gran variedad de algoritmos de aprendizaje automático, específicamente, se identificaron cincuenta y cinco (55). La Figura 2.12 permite visualizar los algoritmos utilizados con mayor frecuencia en relación con las categorías definidas previamente. En consecuencia, se observa que la categoría orientada a la resolución de problemas lingüísticos que derivan del uso del lenguaje natural en documentos de requerimientos y artefactos (NLI), junto a la clasificación de requerimientos de software (SRC) han experimentado de manera activa con diversos algoritmos de aprendizaje automático siendo esto esencial para obtener madurez en el área.

2.6.3/ Aplicación del Aprendizaje Automático Supervisado en la Ingeniería de Requerimientos

Debido a que los artículos analizados no indican con exactitud las actividades de la IR que podrían abordarse mediante los nuevos enfoques y propuestas, en esta investigación se asociaron las categorías previamente definidas (siendo estas meramente los objetivos generales que facilitan la comprensión de los estudios recuperados y analizados) con las actividades correspondientes al proceso de la IR. Esto es útil para identificar tendencias respecto al uso de los métodos de aprendizaje automático en el área.

Cabe señalar que para realizar dicha asociación se consideraron las actividades de *obtención*, *análisis*, *especificación* y *validación* de requerimientos, sugeridas por Wiegers y Hokanson (2023). Asimismo, las subtareas involucradas en cada actividad que conforman el proceso de la IR también contribuyeron con la propuesta (Sommerville, 2011; Wiegers y Hokanson, 2023). La Tabla 2.1 muestra las asociaciones resultantes.

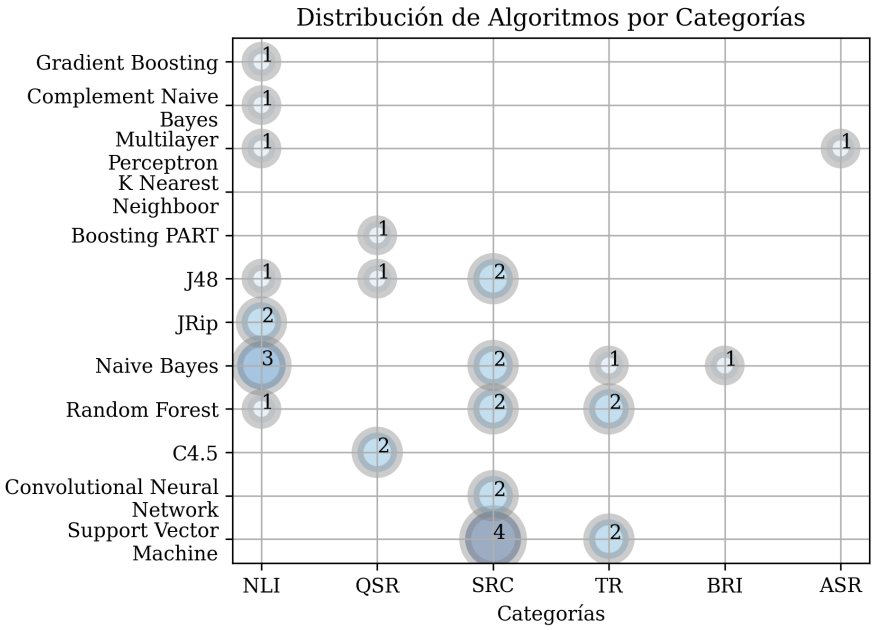


Figura 2.12: Distribución de algoritmos de aprendizaje automático por categorías.

Tabla 2.1: Mapeo entre Categorías y Actividades de Requerimientos.

Actividades de la Ingeniería de Requerimientos				
Categorías	Educción	Análisis	Especificación	Validación
Lenguaje Natural	X			
Clasificación de Requerimientos		X		
Trazabilidad			X	
Reglas de Negocio			X	
Calidad de Requerimientos				X
Análisis de Factibilidad		X		

Luego, al observar la Figura 2.13, se ha determinado que las actividades de la IR de mayor interés y experiencia respecto a la utilización de los algoritmos de aprendizaje automático corresponden al análisis y especificación de requerimientos, respectivamente.

En cuanto a los algoritmos, se destaca que *Naive Bayes* es ampliamente utilizado durante la actividad de obtención de requerimientos. Por otro lado, en el análisis de requerimientos, a menudo se utilizan los algoritmos *Support Vector Machine*. Mientras que, en la actividad de especificación de requerimientos, *Naive Bayes*, *Support Vector Machine* y *Random Forest* son aplicados con mayor frecuencia. Por último, durante la validación de requerimientos, las propuestas identificadas hacen uso del algoritmo *C 4.5*. Por último, se observa que el análisis de requerimientos es la actividad de la IR que presenta una variedad superior de algoritmos aplicados, en contraste a las otras mencionadas anteriormente.

En resumen, los algoritmos más utilizados corresponden a *Naive Bayes* y *Support Vector*

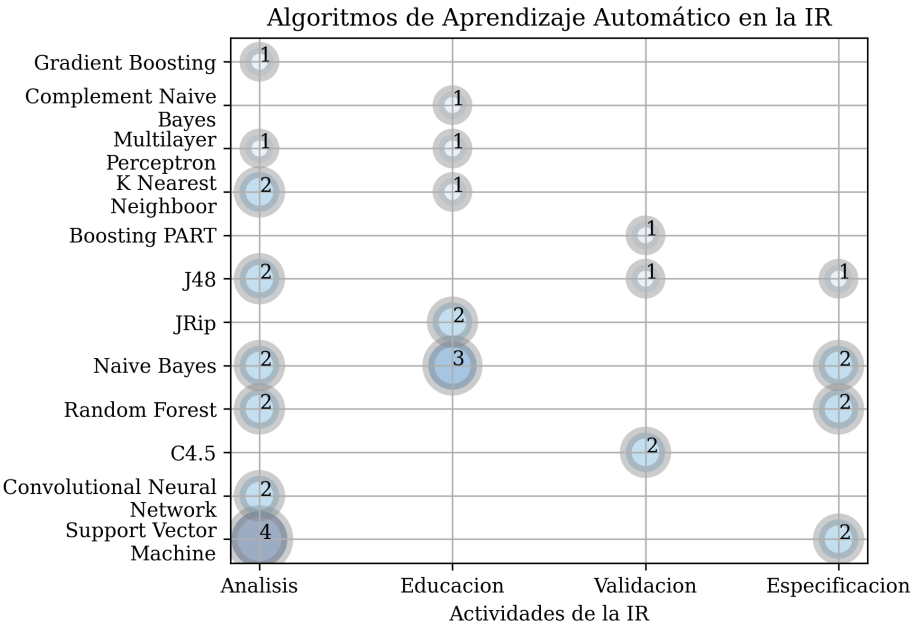


Figura 2.13: Algoritmos de aprendizaje automático aplicados en la IR.

Machine. El algoritmo *Naive Bayes* ha sido utilizado para abordar desafíos relacionados a la clasificación de requerimientos, detección de problemas lingüísticos, trazabilidad y predicción de fallas. Por su parte, el algoritmo *Support Vector Machine* es aplicado en propuestas orientadas a la clasificación de requerimientos funcionales y no funcionales, así como para el agrupamiento del contenido de documentos y artefactos de requerimientos e incluso, es usado en la trazabilidad.

Aún más, resulta clave destacar que los estudios analizados no proporcionan una justificación respecto a los algoritmos aplicados. Además, se detectó que aproximadamente el 15% de los estudios identificados utilizaron más de un algoritmo en sus propuestas de investigación, con la intención de seleccionar aquel que retorna mayor rendimiento. Esto denota la necesidad de un método formal que sirva de guía al proceso de selección de algoritmos de aprendizaje automático para ser aplicados en el área de IR, según el tipo de actividad que se desea abordar.

2.7/ Conclusiones

En este capítulo se presenta el concepto de calidad en la IR. Especialmente, se destaca la importancia de la obtención de requerimientos de calidad y su reconocimiento como factor determinante de éxito en un proyecto de desarrollo de software. Luego, se señalan los principales desafíos que surgen al evaluar la calidad de los requerimientos debido al uso del lenguaje natural en las actividades de obtención.

Más adelante, se revisan las características de calidad provistas por el estándar ISO IEC IEEE

29148 (2018), documento de referencia para el desarrollo de esta investigación. Por último, este capítulo presenta un análisis exploratorio de la literatura acerca de la integración de la IA en la IR, especialmente el aprendizaje automático supervisado, a fin de evidenciar su aplicación a los desafíos presentes en el área. En consecuencia, el análisis de los estudios seleccionados revela que las técnicas de aprendizaje automático supervisado pueden ser efectivamente aplicadas para resolver y automatizar tareas relacionadas con la IR, enfocándose principalmente en la resolución de problemas lingüísticos que derivan del uso del lenguaje natural en documentos de requerimientos y artefactos, clasificación de requerimientos, trazabilidad, identificación de reglas de negocio, análisis de factibilidad y calidad de los requerimientos.

Entre las actividades más abordadas que constituyen el proceso de la IR se identifican el análisis y especificación de requerimientos. Otro aspecto para destacar es la gran variedad de algoritmos aplicados en las propuestas de investigación, lo que denota la necesidad de un procedimiento o consideraciones formales que faciliten el proceso de selección de éstos para su aplicación en el área de requerimientos. Los resultados obtenidos, además, permiten evidenciar el valor de las tecnologías de la IA en el área de la IR ya que, tal como demuestran las propuestas analizadas, éstas permiten la optimización y automatización de las actividades involucradas en el desarrollo de software.

Si bien, no se encontraron trabajos que desestimen la aplicación de las técnicas de aprendizaje automático en la obtención de mejoras y resolución de problemas en dicha área de investigación. Resulta importante señalar que las propuestas analizadas durante la revisión de la literatura, no logran capturar información contextual a partir de expresiones textuales en lenguaje natural mediante las técnicas de aprendizaje automático aplicadas.

Esto representa un limitante para el análisis y evaluación de la calidad de los requerimientos propuesto en este trabajo de Tesis. De hecho, en el procesamiento de lenguaje natural, el contexto es fundamental para comprender el significado de las palabras y oraciones en un documento.

Para abordar esta situación se propone el uso de las redes neuronales recurrentes (RNNs) y redes neuronales transformadoras. Ambas arquitecturas son novedosas en el sentido de que han logrado avances significativos en el procesamiento de lenguaje natural. Las RNNs introdujeron el concepto de memoria recurrente, permitiendo a las redes neuronales capturar información de contexto y estructuras secuenciales (Mikolov et al., 2010). Por otro lado, las redes neuronales transformadoras revolucionaron dicha área al introducir los mecanismos de atención (Brașoveanu y Andonie, 2020).

El siguiente Capítulo 3 aborda el empleo de estas arquitecturas para resolver los desafíos relacionados a la evaluación de la calidad de los requerimientos de software.

2.8/ Summary

In this chapter, the concept of quality in Requirements Engineering (RE) is presented. Specifically, it focuses on the importance of obtaining high-quality requirements and their recognition as a determining factor for success in software development projects. The main challenges that arise when evaluating requirement quality due to the use of natural language in the acquisition activities are then highlighted.

Furthermore, the quality characteristics provided by the ISO IEC IEEE 29148 (2018) standard, which serves as a reference for this research, are reviewed. Lastly, this chapter presents an exploratory analysis of the literature regarding the integration of Artificial Intelligence (AI) in RE, particularly supervised machine learning, to demonstrate its application to the challenges in the field. Consequently, the analysis of selected studies reveals that supervised machine learning techniques can effectively be applied to solve and automate tasks related to RE. These techniques primarily focus on resolving linguistic problems arising from the use of natural language in requirement documents and artifacts, such as requirement classification, traceability, identification of business rules, feasibility analysis, and requirement quality analysis.

Among the most addressed activities in the RE process are requirement analysis and specification. Another aspect worth noting is the wide variety of algorithms applied in the research proposals, indicating the need for a formal procedure or considerations to facilitate the selection process for their application in the requirements field. Additionally, the obtained results highlight the value of AI technologies in the RE field, as demonstrated by the analyzed proposals, which enable optimization and automation of activities involved in software development.

While no studies were found that discourage the application of machine learning techniques for improvement and problem-solving in this research area, it is important to note that the analyzed proposals during the literature review were unable to capture contextual information from textual expressions in natural language using the applied machine learning techniques. This represents a limitation for the analysis and evaluation of requirement quality proposed in this thesis. In natural language processing, context is crucial for understanding the meaning of words and sentences in a document. To address this situation, the use of recurrent neural networks (RNNs) is proposed. RNNs have a hidden state that enables them to maintain internal memory and capture relevant information from the input sequence's history. This allows them to make connections between previous and subsequent words and sentences, facilitating the understanding of sentence meaning in context. In other words, this state enables them to remember the previous context as the text is being processed.

In recent years, RNNs have been predominantly applied in various tasks, such as sentiment analysis, speech recognition, and image processing, due to their ability to process sequential information. RNNs are considered suitable for modeling context dependencies present in language and tasks related to sequence modeling. Another favorable characteristic for their use is the ability to handle variable text length (Mikolov et al., 2010). As a result, RNNs are

well-suited for addressing the processing and analysis of software requirements expressed in natural language for quality study.

On the other hand, this thesis also addresses the application of transformer neural networks for studying requirement quality. Transformer neural networks excel at capturing long-range relationships in text, allowing them to understand context and dependencies throughout a sequence (Brașoveanu y Andonie, 2020). Unlike RNNs, which process sequences sequentially, transformers employ a parallel attention structure that enables each element of the sequence to relate to all other elements, regardless of their relative position. This parallel attention structure allows for capturing long-range dependencies and modeling complex contextual relationships in text, resulting in better performance in natural language processing tasks.

Both architectures are innovative in the sense that they have made significant advancements in natural language processing. RNNs introduced the concept of recurrent memory, enabling neural networks to capture context information and sequential structures. On the other hand, transformer neural networks revolutionized the field by introducing attention mechanisms.

Based on the aforementioned, the following Chapter 3 addresses the use of these architectures to tackle the challenges related to evaluating software requirement quality.

3

Modelos de Clasificación Binaria para Evaluar la Calidad de los Requerimientos

Las tendencias cambiantes en la industria del software aceleran la evolución de sus procesos de desarrollo. Principalmente, la integración de tecnologías de vanguardia como la Inteligencia Artificial ha generado oportunidades sin precedentes para explorar nuevas técnicas y ajustar métodos de desarrollo, además de permitir la automatización de tareas que demandan esfuerzos intensivos, con el fin de proveer soluciones que contribuyan a mejorar la calidad del software. En este sentido, el presente capítulo aborda la aplicación de técnicas tradicionales de procesamiento de lenguaje natural junto a las redes neuronales para llevar a cabo el estudio de la calidad de los requerimientos de software. En efecto, se presenta un conjunto de modelos de clasificación binaria basados en aprendizaje profundo con el propósito de evaluar las características de calidad de requerimientos expresados en lenguaje natural. En primer lugar, se describe el escenario que motiva el desarrollo de esta propuesta. Luego, se presenta la metodología de trabajo utilizada en la construcción de cada modelo neuronal. Finalmente, se provee un análisis comparativo sobre los resultados obtenidos.

3.1/ Introducción

En la literatura es posible hallar una amplia variedad de estrategias para abordar los desafíos asociados a la calidad de los requerimientos cuando éstos se especifican mediante el uso del lenguaje natural, los cuales han sido detallados previamente en el Capítulo 2. Las estrategias incluyen: *Sistemas basados en Reglas*, *Modelado y Formalización* y, por último, *Clasificación Automática*. En contraste al capítulo anterior que tiene por objeto evidenciar la aplicación de la Inteligencia Artificial (IA) en la Ingeniería de Requerimientos (IR) y determinar su factibilidad de uso en el área, la siguiente sección se enfoca sólo en analizar los antecedentes más difundidos que aplican técnicas de aprendizaje automático para resolver aspectos relacionados a la calidad de los requerimientos. El propósito de este análisis es determinar las principales fuentes de

datos, técnicas de procesamiento y arquitecturas frecuentemente utilizadas.

3.1.1/ Enfoques Automáticos para Evaluar la Calidad de los Requerimientos

Las propuestas basadas en técnicas de aprendizaje automático se centran en las tareas de preprocesamiento para obtener etiquetas gramaticales, bolsas de palabras y determinar la co-ocurrencia de términos. Luego, con ello se obtienen representaciones adecuadas para alimentar los algoritmos y determinar la precisión. Las investigaciones más difundidas que adoptan como estrategia la aplicación de técnicas de aprendizaje automático se sintetizan a continuación.

Halim y Siahaan (2019) abordan el estudio de la *atomicidad* de los requerimientos de software, condición reemplazada por la característica de calidad *singular* en el estándar vigente. En su investigación, se compara el rendimiento de 3 (tres) algoritmos de aprendizaje automático, Bayes Net, Random Forest y Multilayer Perceptron. Primero, realizan el preprocesamiento de datos que consiste en el etiquetado gramatical, eliminación de palabras de parada (stopwords) y lematización de palabras. Después, los algoritmos se entrenan con un corpus de requerimientos reducido (600 requerimientos) y fuertemente desbalanceado. Los autores señalan que el 78% del conjunto de datos corresponde a requerimientos etiquetados como atómicos, por lo que representan la clase mayoritaria. Los resultados reportados demuestran que Bayes Net retorna 77% de precisión y tiene un rendimiento superior respecto a los otros algoritmos.

Mezghani et al. (2018) proponen un enfoque automático que utiliza el algoritmo K-means para detectar redundancias e inconsistencias que conducen a la definición de requerimientos *ambiguos*. Durante el preprocesamiento se realiza el etiquetado gramatical y la extracción de fragmentos de sustantivos (*noun chunking*). Esta última es utilizada por los autores para identificar patrones sintácticos que refieren a términos de índole técnico que con frecuencia se emplean para definir requerimientos. Luego, se aplica el algoritmo K-means para agrupar los requerimientos de acuerdo con su similitud utilizando como métrica la distancia euclidiana. Sin embargo, los resultados señalan que aún no es posible identificar de forma correcta redundancias e inconsistencias en clusters con información contextual similar, dado que no siempre es posible captar la relación semántica entre las palabras.

Osman y Zaharin (2018) analizan la *ambigüedad* de los requerimientos. Para ello, proponen un análisis comparativo en el que se consideran 9 (nueve) algoritmos de clasificación (OneR, Naive Bayes, Logistic Regression, K-Nearest Neighbor, Decision Table, Decision Stump, J48, Random Forest y Random Free). El conjunto de datos utilizado se compone de 180 requerimientos. El preprocesamiento se realiza de forma manual y consta de la eliminación de palabras de parada (stopwords) y lematización de palabras y, además, las abreviaciones son transformadas a su palabra original. Los autores señalan que el algoritmo que retorna mejores resultados es Random Forest. A pesar de esto, también destacan que presenta dificultades en la generalización de la predicción sobre nuevos requerimientos, debido al reducido tamaño del

conjunto de datos utilizado durante el entrenamiento.

Ormandjieva et al. (2007) en su investigación proponen la detección de requerimientos ambiguos mediante la aplicación de Decision Tree. El análisis de la ambigüedad se realiza en base a dos perspectivas: *comprensión superficial* y *comprensión conceptual*. En el nivel de comprensión superficial pueden intervenir varios factores, tales como longitud de las frases, adjetivos y adverbios ambiguos, verbos pasivos, entre otros. Mientras que en la comprensión conceptual se incluye el análisis de la presencia de ruido, contradicciones, vínculos lógicos entre hechos o acontecimientos, entre otros.

En cuanto al conjunto de datos, éste se compone de 165 requerimientos. La distribución de los datos para la clase de comprensión superficial presenta el 92.7% de requerimientos no ambiguos. En cuanto a la comprensión conceptual, ésta consta de 83.6% requerimientos no ambiguos. En ambos casos, la clase mayoritaria está representada por requerimientos no ambiguos. A partir de ello, se puede deducir que la muestra no es uniforme. Por lo tanto, en esta propuesta se destacan dos aspectos importantes: primero, datos desbalanceados y luego, bajo acuerdo promedio en el proceso de etiquetado. Este último, indica que los involucrados en el etiquetado de datos coinciden en sólo el 65% de los requerimientos clasificados. A pesar de ello, los resultados obtenidos señalan 86% de precisión en lo que refiere a la ambigüedad superficial. Por otro lado, señalan que no se hallaron indicadores objetivos que permitan abordar de forma automática la ambigüedad relacionada con la comprensión conceptual.

Yang et al. (2010b) también presentan una herramienta para facilitar la detección de la ambigüedad. Principalmente, se focalizan en distinguir requerimientos con ambigüedad nociva e inocua. El conjunto de datos utilizado sólo presenta el 12% de requerimientos ambiguos. En lo que respecta al preprocesamiento, se realiza la segmentación de palabras en unidades más pequeñas (tokens) y el etiquetado gramatical. El propósito de la herramienta propuesta es detectar construcciones ambiguas y reconocer las entidades que las originan. Para ello, se aplican heurísticas definidas manualmente que determinan la co-ocurrencia de términos y similitud semántica. En combinación con el algoritmo de aprendizaje automático LogitBoost, se identifican los requerimientos con ambigüedad considerando las dos clases mencionadas, nociva e inocua. Los autores señalan, además, la definición de un umbral de ambigüedad asociado al porcentaje de acuerdo entre los involucrados en el proceso de clasificación de requerimientos, el cual corresponde al 70%. Por último, los resultados reportados indican un 75% de precisión.

Posteriormente, Yang et al. (2012) proponen un enfoque basado en reglas junto con el algoritmo Campos Aleatorios Condicionales para detectar incertidumbre y frases especulativas que pueden conducir a la definición de requerimientos *incompletos* y *ambiguos*. El propósito es identificar las señales de incertidumbre y determinar automáticamente su alcance sobre la frase bajo análisis. En otras palabras, indicar si la frase es «especulativa» o «no especulativa» y determinar los fragmentos que resultan afectados. Para llevar a cabo el preprocesamiento de las frases se realiza la segmentación de palabras en tokens y lematización. Un aspecto clave de esta

investigación es que los autores mencionan la necesidad de acceder a información contextual para identificar señales de incertidumbre. Para ello, adoptan como estrategia considerar 3 (tres) palabras vecinas ubicadas antes y después de cada palabra. Otro aspecto que se considera son las relaciones de dependencia sintáctica entre las etiquetas gramaticales y la co-ocurrencia como indicadores de presencia de incertidumbre. Sin embargo, los resultados obtenidos señalan dificultades en la identificación de frases especulativas.

Parra et al. (2015) proponen evaluar la calidad de un conjunto de características de calidad. Específicamente se abordan las características de calidad: verificable, modificable, validable, consistente, completo, no ambiguo, comprensible, trazable, abstracto, preciso y atómico. Para ello, presentan un clasificador binario que utiliza los algoritmos C4.5, PART, Boosting y Bagging. El clasificador es binario porque sólo dispone de dos etiquetas para indicar buena o mala calidad. La estrategia de representación de requerimientos que se utiliza para alimentar a los algoritmos es poco convencional, dado que sólo se basa en valores métricos. En otras palabras, primero los requerimientos se evalúan mediante métricas definidas previamente y luego, los resultados obtenidos se utilizan para generar representaciones vectoriales sin focalizar en la extracción de características e información contextual. El conjunto de datos está compuesto por 1035 requerimientos y, en contraste a otras investigaciones, sus clases se distribuyen de manera uniforme. Los autores señalan 545 requerimientos incluidos en la clase buena calidad y 490 en mala calidad. En cuanto a los resultados reportados, el algoritmo que retorna mayor rendimiento es Bagging con el 87% de precisión.

Después de este análisis, es posible reconocer los métodos de preprocesamiento, arquitecturas y conjuntos de datos frecuentemente utilizados para abordar la calidad de los requerimientos. La Tabla 3.1 resume los principales enfoques automáticos identificados en el estudio de la calidad.

Tabla 3.1: Síntesis de enfoques automáticos.

Característica	Fuente Datos	Algoritmo	Preprocesamiento
Singular	ACM's OOP-SLA DesignFest	Bayes Net Random Forest Multilayer Perceptron	Tokenización POS Tagging Stopwords Lematización
No Ambiguo	Industria	K-Means	Tokenización POS Tagging Noun chunking
	Propia	OneR Naive Bayes Logistic Regression K-Nearest Neighbor Decision Table Decision Stump J48 Random Forest Random Tree	POS Tagging Co-ocurrencia de términos
	Académico	Decision Tree C4.5	Tokenización POS Tagging Análisis sintáctico
	Industria	LogitBoost	Tokenización POS Tagging
No Ambiguo Completo	RE@UTS web pages	Conditional Random Field	Tokenización Lematización Co-ocurrencia de términos
Verificable Modificable Validable Consistente Completo No Ambiguo Comprensible Trazable Abstracto Preciso Atómico	INCOSE	C4.5 PART Boosting Bagging	Representaciones basadas en métricas

3.1.2/ Limitaciones

Si bien las propuestas anteriores constituyen un precedente valioso para la IR respecto a la evaluación y control de calidad, se advierten ciertas limitaciones que se listan a continuación:

- Dificultades para representar y extraer información contextual a partir de requerimientos expresados en lenguaje natural.
- Algunos autores advierten que la estimación de la calidad de los requerimientos es una tarea subjetiva, por lo que se requiere definir umbrales o acuerdos promedios respecto al criterio de evaluación de los datos que luego se utilizarán para entrenar los algoritmos.
- Entrenamiento de algoritmos con conjuntos de datos fuertemente desbalanceados.
- Incorporación de heurísticas definidas manualmente para reforzar el análisis sintáctico y semántico.
- Los escasos repositorios públicos de requerimientos conducen a la construcción de conjuntos de datos de autoría propia, que resultan manualmente costosos.
- No es posible acceder a los conjuntos de datos a través de los enlaces provistos por las investigaciones.
- Evaluación parcial de la calidad, dado que la mayoría de las investigaciones abordan el análisis de las características de calidad individuales o en su defecto, un reducido conjunto de características.
- La ambigüedad es la característica de calidad que presenta mayor madurez en cuanto a la aplicación de técnicas de aprendizaje automático para su evaluación. La madurez se refiere a la capacidad de estas técnicas para abordar de manera confiable y precisa la complejidad de los diferentes tipos de ambigüedad que pueden presentarse en el lenguaje natural.

3.1.3/ Escenario Motivador

La literatura evidencia que llevar a cabo controles de calidad de forma automática sobre requerimientos expresados en lenguaje natural no es una tarea trivial. Afortunadamente, los avances en el campo de la IA permiten experimentar y evaluar nuevas soluciones. Por esta razón, en este trabajo de Tesis se presentan dos arquitecturas de aprendizaje profundo que permiten abordar los desafíos que caracterizan al lenguaje natural: (i) *redes neuronales recurrentes* y las (ii) *redes neuronales transformadoras*. Ambos enfoques poseen mecanismos para tratar la naturaleza secuencial inherente del lenguaje. Las redes neuronales recurrentes (RNNs) ofrecen una nueva forma de representar el contexto de documentos y/o frases, en consecuencia, permite que el modelo considere la información contextual que proviene de palabras temporalmente distantes para generar predicciones. Por su parte, las redes neuronales transformadoras ofrecen nuevos mecanismos, *autoatención* y *codificadores posicionales*, que ayudan a representar el tiempo y focalizar en las relaciones entre palabras, a pesar de su distancia temporal. Por ello, se propone el uso de estas potentes arquitecturas para obtener un conjunto de modelos de clasificación binaria y, con ello, lograr determinar de forma automática el cumplimiento de las características de calidad de los requerimientos de software.

En resumen, esta propuesta -en contraste a las existentes en la literatura- permite procesar los requerimientos considerando la naturaleza secuencial del lenguaje. Esto es posible dado que los modelos propuestos se basan en redes neuronales robustas y representaciones vectoriales densas capaces de acceder computacionalmente a palabras distantes en el tiempo, que permiten considerar el contexto de una palabra en particular. Este mecanismo permite capturar las relaciones sintácticas y semánticas entre palabras, y así obtener información contextual de los requerimientos. En consecuencia, esto permite a los modelos basados en redes neuronales determinar el cumplimiento de las características de calidad con mayor eficiencia.

3.2/ Metodología

La metodología de trabajo utilizada para la construcción de los modelos de clasificación binaria basados en redes neuronales sigue los lineamientos propuestos por Amershi et al. (2019). La Figura 3.1 proporciona una visión general de las fases que componen la metodología de trabajo. Ésta consta de 9 (nueve) fases, algunas de ellas orientadas a la gestión de datos (recolección, limpieza y etiquetado), *configuración de los modelos* (requerimientos, ingeniería de características, entrenamiento y evaluación) y *despliegue de cada modelo* (implementación y monitoreo). El flujo de trabajo también incluye una serie de bucles de retroalimentación que permiten realizar ajustes sobre los modelos. Aquellos bucles de retroalimentación en rojo indican que a partir de las fases de evaluación y monitoreo es posible retroceder a cualquier fase anterior para realizar ajustes. Mientras que el bucle de retroalimentación en azul indica que desde la fase de entrenamiento es posible retroceder a la fase de ingeniería de características para llevar a cabo cambios pertinentes.

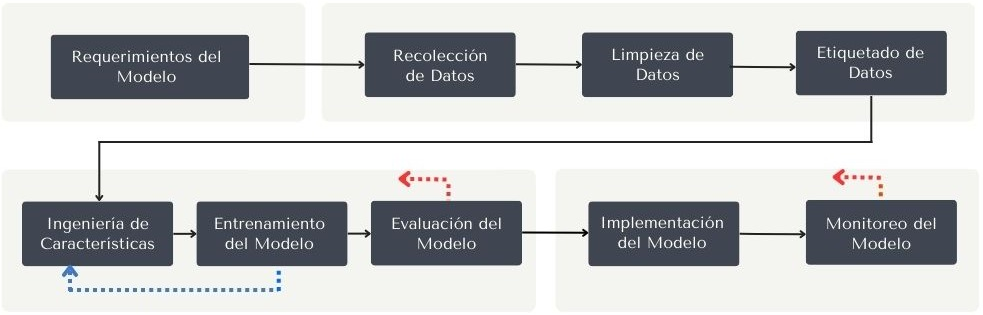


Figura 3.1: Metodología para la construcción de modelos neuronales.

3.2.1/ Fase Requerimientos del Modelo

La fase *Requerimientos del Modelo* consiste en la selección de los algoritmos de aprendizaje (clustering, clasificación, regresión, entre otros) más apropiados para el problema que se desea resolver. Además, se discuten aspectos tales como el enfoque de aprendizaje que se adoptará (supervisado, no supervisado, por refuerzo) y las características factibles de ser analizadas e implementadas mediante la aplicación de estas técnicas. Particularmente, en este trabajo de Tesis se aplica aprendizaje profundo, redes neuronales recurrentes (RNNs) y redes transformadoras que incluyen un enfoque de aprendizaje del tipo supervisado para el estudio de la calidad de los requerimientos de software. A continuación, se analizan las principales características de ambos tipos de redes, las cuales promueven su uso en tareas relacionadas con el procesamiento de lenguaje natural y que permiten justificar su aplicación en esta investigación.

3.2.1.1/ Redes Neuronales Recurrentes

Las RNNs se caracterizan por su capacidad de procesamiento de información secuencial. En este tipo de redes neuronales, una secuencia de entrada se representa con un vector de tamaño fijo que mediante tokens alimenta a una unidad recurrente. El término recurrente implica que se realiza el mismo cálculo sobre cada token que compone la secuencia de entrada, además que cada siguiente paso en el tiempo (*time step*) depende de los resultados anteriores. En cierto modo, las RNNs tienen

memoria sobre los cálculos anteriores y utilizan esta información en el procesamiento actual. Dado que las RNNs realizan el procesamiento secuencial modelando unidades en secuencia, tienen la capacidad de capturar la naturaleza secuencial inherente presente en el lenguaje natural, donde las unidades representan caracteres, palabras u oraciones. Puesto que las palabras de un lenguaje desarrollan su significado semántico en función de las palabras anteriores que componen una oración, las RNNs resultan convenientes para modelar tales dependencias de contexto en el lenguaje y tareas similares de modelado de secuencias. Otro factor que favorece el uso de las RNNs para tareas de modelado de secuencias radica en su capacidad para modelar longitudes variables de texto, incluyendo oraciones, párrafos e incluso documentos.

A diferencia de otras arquitecturas, las RNNs tienen pasos computacionales (*time steps*) flexibles que brindan una mejor capacidad de modelado y permiten capturar un contexto ilimitado (Young et al., 2018). En consecuencia, esto ha resultado una fuerte motivación para los investigadores respecto a su uso en diversos dominios de aplicación relacionados con el procesamiento del lenguaje natural. Por lo expuesto, es que este trabajo de Tesis plantea inicialmente el uso de las RNNs para abordar el procesamiento y análisis de los requerimientos de software en lenguaje natural y, con ello, determinar su calidad.

3.2.1.2/ BERT: Representación de Codificador Bidireccional de Transformadores

Uno de los mayores desafíos en el procesamiento del lenguaje natural mediante la aplicación de técnicas de aprendizaje profundo es la escasez de datos de entrenamiento. Debido a que el procesamiento del lenguaje natural es un campo de estudio con múltiples aplicaciones, la mayoría de los conjuntos de datos disponibles para tareas específicas sólo contienen unos pocos ejemplos de entrenamiento etiquetados. Sin embargo, el procesamiento del lenguaje natural basado en modelos de aprendizaje profundo sólo resulta útil cuando se utilizan grandes volúmenes de datos. Con el propósito de abordar este desafío, surgieron diversas técnicas, tales como el aprendizaje por transferencia (*transfer learning*), que permiten entrenar modelos con una enorme cantidad de datos sin etiquetar extraídos de la web. Luego, esto permite la generación de modelos previamente entrenados de propósito general, que pueden ser ajustados (*fine tuning*) para su posterior uso en tareas específicas relacionadas con el procesamiento de lenguaje natural. Particularmente, lo que motiva el uso de modelos previamente entrenados es la búsqueda de una mejora sustancial en el rendimiento de las tareas que comprenden procesamiento del lenguaje natural.

En relación con lo anterior, este trabajo de Tesis también incluye la evaluación de un nuevo modelo de representación de lenguaje denominado Bidirectional Encoder Representations from Transformers (BERT por sus siglas en inglés) para el estudio de la calidad de los requerimientos (Kenton y Toutanova, 2019). En contraste a otros modelos de representación de lenguaje, BERT está diseñado para pre-entrenar representaciones bidireccionales profundas a partir de texto no etiquetado, considerando conjuntamente el contexto izquierdo y derecho en todas las capas. De este modo, es capaz de analizar las palabras que se encuentran tanto a la izquierda como a la derecha de una palabra clave, y esto le permite entender en profundidad el contexto. Como resultado, el modelo BERT pre-entrenado se puede ajustar con sólo una capa de salida adicional para crear modelos de última generación y, con ello, llevar a cabo una amplia gama de tareas asociadas al procesamiento de lenguaje natural.

3.2.2/ Fase Recolección de Datos

El propósito de la fase *Recolección de Datos* es identificar conjuntos de datos que puedan ser utilizados para entrenar los modelos de aprendizaje automático. En esta fase es posible generar conjuntos de datos propios o fusionar conjuntos de datos provenientes de diversas fuentes para trabajar sobre ellos. De hecho, esta fase es reconocida por su criticidad e influencia absoluta en el rendimiento de los modelos. Esto es así dado que, si un modelo se alimenta con datos de entrada inadecuados, generará efectos adversos en las inferencias resultantes. Los datos en esta fase de recolección se almacenan en su formato original, sin ningún tipo de preprocesamiento previo, por lo que su uso y análisis aún no son factibles mediante las técnicas de aprendizaje automático. En la siguiente sección se describe el corpus de requerimientos resultante de la fase de recolección de datos, que se utiliza para entrenar los modelos neuronales propuestos.

3.2.2.1/ Análisis Descriptivo del Corpus de Requerimientos

A fin de proveer los datos necesarios para el entrenamiento de los modelos neuronales, se conformó un corpus compuesto por 758 requerimientos extraídos de *Public Requirement Dataset* (PURE) (Ferrari et al., 2017), y 142 requerimientos provenientes del conjunto propuesto por Saavedra (2016) en su trabajo de Tesis. Por lo tanto, se llevó a cabo la fusión de dos conjuntos de datos provenientes de diferentes fuentes, con el objetivo de enriquecer el corpus de requerimientos utilizado por los modelos neuronales. Ambos forman parte del reducido conjunto de datos públicos de requerimientos de software disponibles en la literatura. La disponibilidad de conjuntos de datos públicos constituye uno de los principales desafíos que debe abordar la comunidad de la IR al integrar técnicas que derivan de la IA en sus investigaciones (Gramajo et al., 2020a).

El conjunto de datos PURE proporciona 79 documentos de requerimientos en lenguaje natural extraídos de la web. Los documentos contenidos en PURE están escritos en inglés y cubren múltiples dominios procedentes de diversas compañías y proyectos universitarios. Del mismo modo, los requerimientos extraídos del segundo conjunto están redactados en inglés y refieren al dominio bancario, específicamente sistemas de Home Banking. La heterogeneidad de los requerimientos incluidos en el corpus construido permite evaluar la capacidad de generalización de los modelos propuestos. Los requerimientos fueron seleccionados de manera aleatoria y extraídos manualmente. Sólo se incluyeron requerimientos expresados en texto plano y se excluyeron aquellos definidos como enlaces a documentos externos. También se excluyeron los requerimientos contenidos en figuras y tablas. La Tabla 3.2 sintetiza el corpus de requerimientos. Particularmente, se señala dominio y cantidad de requerimientos extraídos.

Tabla 3.2: Corpus de Requerimientos.

Dataset	**Dominio**	**Requerimientos**
PURE	SmartHome	14
	Education	56
	Movies	25
	Games	31
	ATM	142
	Management	469
	Law	40
	Health	10
	E-commerce	45
	Cars	62
	Stock	15
Saavedra (2016)	Automated Teller Machine	133

3.2.3/ Fase Limpieza de Datos

En la fase *Limpieza de Datos* se remueven o corrigen aquellos registros con valores dañados o no válidos causados por errores en la entrada o medición de los datos (valores atípicos, valores perdidos, redundancia de información). Para ello, existen diversas técnicas que se pueden aplicar, tales como la transformación de distribuciones, creación de nuevas características, estrategias para reemplazar valores perdidos, compresión de datos, delimitar datos anómalos, entre otras.

Particularmente en esta investigación, los ruidos y anomalías atribuibles a los estilos de escritura, la sintaxis y/o la ortografía no fueron eliminados, dado que son necesarios para el análisis de la calidad percibida por cada modelo neuronal. En relación a esto, los ruidos identificados en el corpus de requerimientos corresponden a la inserción de símbolos y caracteres adicionales, ausencia e inclusión de espacios entre palabras, repetición de palabras, utilización de abreviaturas y acrónimos. La Tabla 3.3 resume y ejemplifica los ruidos identificados en el conjunto de datos, en base a la taxonomía propuesta por Al Sharou et al. (2021).

Tabla 3.3: Tipos de Ruidos en el Corpus de Requerimientos.

Tipo de Ruido	Descripción de Requerimiento
Símbolos	A list of all disputes cases that meet the entered search criteria must be provided to the user. The list of cases must differentiate the type and/or status of the case by color -coding the line item in the list. For example all open cases will be yellow and all closed cases will be gray.
	Support for individual and shared staff login accounts; access to modules is granted by use of «roles» or «privileges» that allow each account to access as many (or as few) modules as needed. Individual logins allow user-level preferences and audit trail.
Puntuación	The user can change his account information (eg. password, e-mail address ,. . .).
	The administrator shall have the ability to block all access to the system for all users or selectively by user. (All blocked users with active sessions shall automatically be logged off.)
Repetición de palabras	Integrate the full-text search (search-tools project) module with the WARC browser to provide users with WARC indexing/searching capabilities.
Significado Oculto	The Thermal Control System shall update the display of Current System Temperature every 10 +/- 1 seconds.
Acrónimos	Ability for multiple staff members and patrons to simultaneously access and update patron and item records, including on staff check-in and check-out terminals, on self check-out stations, through SIP2/NCIP2 and similar protocols and APIs, and in OPAC.
Ortografía	A user can extend his Search by filling in an additional form, where they can specify one (or more) keyword(s) of the Publication and/or by specifying the Pub-lications that refer to it.

3.2.4/ Fase Etiquetado de Datos

En la fase *Etiquetado de Datos*, tal como su nombre lo indica, cada registro contenido en el conjunto de datos es etiquetado. En consecuencia, a aquellos registros correspondientes a una misma clase se les asigna una etiqueta idéntica. Una clase puede ser definida como un conjunto de datos con cierto grado de similitud o relación. La fase de etiquetado de datos es reconocida por ser costosa en cuanto a tiempo y trabajo manual asociado. En la siguiente sección se presentan las variables categóricas consideradas durante el etiquetado de requerimientos.

3.2.4.1/ Descripción de Variables Categóricas

El corpus de requerimientos se compone por 900 individuos que representan a los requerimientos de software. Cada requerimiento contenido en el corpus cuenta con un identificador y su respectiva descripción. Además, consta de 9 (nueve) variables categóricas y dicotómicas que corresponden a las características de calidad: *'Necesario'*, *'Apropiado'*, *'No Ambiguo'*, *'Completo'*, *'Singular'*, *'Factible'*, *'Verificable'*, *'Correcto'* y *'Conforme'*. En consecuencia, los requerimientos fueron clasificados considerando sólo dos valores posibles 0 (cero) y 1 (uno) para indicar el cumplimiento de las características de calidad. El proceso de clasificación fue realizado bajo la supervisión de investigadores en el área de la IR. A partir de ello, es posible establecer el corpus de requerimientos que se utilizará durante el entrenamiento de los modelos neuronales. La Tabla 3.4 muestra los primeros 3 (tres) registros del corpus de requerimientos.

3.2.4.2/ Análisis de Distribución de Variables

Luego, se analiza la distribución de frecuencia de las variables en el conjunto de datos. Con ello, se observa que las variables correspondientes a *Característica Apropiado* y *Necesario* están desbalanceadas, es decir, la distribución de frecuencia no es uniforme. La Tabla 3.5 sintetiza los valores correspondientes a la distribución de frecuencia de cada variable incluida en el corpus de requerimientos.

Tabla 3.5: Distribución de frecuencia de características de calidad.

Característica Calidad	**Ocurrencia Positiva**	**Ocurrencia Negativa**
Singular	715	185
Apropiado	888	12
Correcto	363	537
Completo	456	444
Factible	401	499
Verificable	401	499
Conforme	421	479
Necesario	898	2
No Ambiguo	599	301

En esta investigación, se excluye el análisis de la *Característica de Calidad Necesario*. Esta característica se relaciona principalmente con la necesidad de que un requerimiento contribuya a la satisfacción de un objetivo del sistema, que a su vez está determinado por una capacidad o limitación. Por lo tanto, su evaluación a través de modelos neuronales requiere que el corpus de datos esté compuesto por requerimientos provenientes del mismo proyecto, de modo que sea posible analizar las relaciones y dependencias entre ellos, así como considerar la información relacionada con los objetivos del sistema, sus funcionalidades y el contexto en el que opera. Sin embargo, en contraposición a este enfoque, el corpus utilizado en este estudio está formado por requerimientos individuales que provienen de diversos dominios y proyectos. En consecuencia, la evaluación del cumplimiento de la *Característica Necesario* con los datos disponibles en este corpus no resulta factible.

De la misma manera se detectaron sólo 12 (doce) individuos pertenecientes a la clase *No Apropiado* en el corpus de requerimientos. Esto indica que la variable correspondiente a la *Característica de Calidad Apropiado* se encuentra desbalanceada y que la incorporación y/o réplica de más individuos de

Tabla 3.4: Extracto Corpus de Requerimientos.

Id	Descripción	Singular	Apropiado	Correcto	No Ambiguo	Completo	Factible	Verificable	Conforme	Necesario
01	Program Administrators/Nursing Staff Members shall have the ability to modify information relating to a Nursing Student including student ID, student name, phone number, e-mail, status (full time or part time) and program of study.	1	1	1	1	1	1	1	1	1
02	The ATM shall display the Customer Account Number, Account Balance, and so on.	1	1	0	0	0	0	0	0	1
03	The DUAP System shall allow the system administrator to add, modify, and delete Computational Services.	0	1	1	1	1	1	1	0	1

forma intencional puede intervenir en la investigación que se propone. De hecho, es conveniente no incluirla en el análisis, dado que su presencia podría afectar el posterior entrenamiento de los modelos neuronales. En otras palabras, si los datos están desbalanceados la clasificación estaría sesgada hacia la clase predominante. Por este motivo, el estudio de la *Característica Apropiado* tampoco se incluye en este trabajo de Tesis. Por lo tanto, se identifican como características de calidad candidatas para su estudio mediante la aplicación de redes neuronales las correspondientes a *Singular*, *No Ambiguo*, *Conforme*, *Completo*, *Factible*, *Verificable* y *Correcto*.

3.2.5/ Fase Ingeniería de Características

La fase *Ingeniería de Características* involucra las actividades relacionadas a la extracción y selección de características necesarias para los modelos de aprendizaje automático. Esta fase fundamental consiste en realizar una transformación óptima de los datos para que luego, éstos puedan ser procesados mediante algoritmos de aprendizaje automático. De hecho, en esta fase también es posible realizar ajustes sobre los datos de acuerdo a lo requerido por los modelos. Algunos ejemplos de tales ajustes son la normalización, recorte de valores, conversión de variables categóricas mediante técnicas de codificación, entre otros. En la siguiente sección se describe el preprocesamiento realizado sobre los requerimientos a fin de estructurarlos en un formato de entrada conveniente para los modelos propuestos.

3.2.5.1/ Preprocesamiento de Datos

Los requerimientos contenidos en el corpus fueron preprocesados con el propósito de estructurarlos en un formato de entrada adecuado para el entrenamiento de cada modelo neuronal. En primer lugar, se llevó a cabo la segmentación de los requerimientos en tokens. Esto consiste en separar cada requerimiento de entrada en unidades más pequeñas denominadas tokens, reconocidos como componentes básicos del lenguaje natural. Los tokens pueden ser palabras, números, signos de puntuación o símbolos. Los tokens correspondientes a palabras en mayúsculas se normalizaron en minúsculas. Mientras que los tokens que representan símbolos de puntuación fueron eliminados. Luego, los requerimientos representados en vectores de palabras se transformaron en secuencias numéricas, a través del método *word encoding* para su ingreso a la red neuronal. Este método permite mapear las palabras del vocabulario, compuesto por 2028 palabras únicas, a índices escalares.

Otro aspecto para destacar es la longitud variable de los requerimientos que componen el corpus. En este sentido, para cada requerimiento de entrada se estableció como longitud máxima (L) 100 palabras. Esta última ha sido definida en base a la máxima extensión que presentan los requerimientos en el corpus. Para los requerimientos de longitud menor a L, se adoptó la estrategia *pre-sequence padding*. Esta técnica consiste en rellenar con ceros (de izquierda a derecha) las representaciones numéricas hasta alcanzar la longitud establecida. Finalmente, las representaciones numéricas obtenidas se utilizaron para alimentar los modelos neuronales y éstos, mediante un entrenamiento con enfoque de aprendizaje del tipo supervisado, aprendieron a clasificar los requerimientos en función a su calidad (Anexo.1). La Figura 3.2 ilustra el procesamiento de requerimientos.

3.2.6/ Fase de Entrenamiento

En la *fase de entrenamiento*, los modelos se entrenan de forma iterativa hasta alcanzar el rendimiento deseado. Para este propósito, el conjunto de datos se divide en 3 (tres) subconjuntos. En primer lugar,

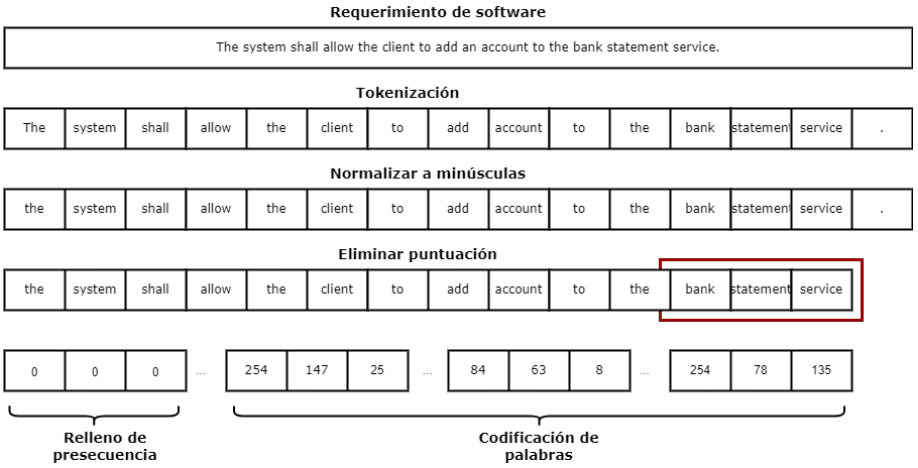


Figura 3.2: Preprocesamiento de requerimientos.

datos de entrenamiento (*training data*), con los cuales se lleva a cabo el entrenamiento de los modelos. Luego, en esta fase también se realizan ajustes sobre los hiperparámetros para mejorar el rendimiento, lo que deriva en diferentes configuraciones de modelos que son evaluados con los datos de validación (*validation data*). Por último, para determinar el rendimiento de cada modelo ya definido, se utilizan los datos de prueba (*testing data*). En la siguiente sección se describe de qué manera se entrenó el conjunto de modelos de clasificación binaria utilizado para determinar la calidad de los requerimientos a través de dos experimentos.

3.2.6.1/ Experimento 1

En esta investigación la evaluación de la calidad de los requerimientos de software es planteada como un problema de clasificación. Por lo que se propone la definición de un conjunto de clasificadores binarios, lo que resulta en un clasificador por cada característica de calidad que se pretende abordar. En consecuencia, se presentan 7 (siete) modelos de clasificación binaria basados en redes neuronales para evaluar la calidad de los requerimientos correspondientes a las características de calidad: *(a) No Ambiguo*, *(b) Verificable*, *(c) Correcto*, *(d) Completo*, *(e) Factible*, *(f) Conforme* y *(g) Singular*. Particularmente, se proponen dos experimentos sobre los datos recolectados. El primero evalúa las características de calidad mediante la aplicación de redes neuronales del tipo recurrentes y luego, un segundo experimento, propone la utilización de redes neuronales transformadoras. El principal propósito es analizar el desempeño de ambas arquitecturas y determinar cuál resulta más adecuada para el estudio de la calidad de los requerimientos expresados en lenguaje natural.

3.2.6.2/ Modelos de Clasificación basados en Redes Neuronales Recurrentes

En esta sección se definen los modelos de clasificación binaria considerando 4 (cuatro) variantes de redes neuronales recurrentes: LSTM, GRU, BiLSTM y BiGRU (Anexo.2). A continuación, se proporcionan detalles sobre la arquitectura de red neuronal propuesta para obtener los modelos de clasificación.

La arquitectura propuesta para cada característica de calidad incluye una capa, *sequence input layer*, que permite el ingreso de los requerimientos representados mediante vectores de índices escalares. Cada

vector de entrada tiene una longitud (L) de 100 palabras, dado que se considera la máxima L disponible en el corpus. Si los requerimientos poseen una longitud menor a L, se rellenan con *zero-padded* para completar el vector. El *input size* para esta capa se establece en uno (1). Luego, se incluye una *word embedding layer*. Esta capa considera como parámetro de entrada el tamaño del vocabulario que se define por las palabras únicas presentes en el corpus de requerimientos. Luego, la *word embedding layer* transforma los requerimientos representados en secuencias de índices para generar un vector denso de M dimensiones por cada palabra del vocabulario. Este procedimiento permite a la red neuronal -cualquiera sea la variante (LSTM, GRU, BiLSTM y BiGRU)- capturar detalles semánticos de las palabras. De modo que, aquellas palabras con significados similares tendrán vectores similares. También permite determinar relaciones entre palabras a través de la aritmética vectorial. Por último, la dimensión de salida para esta capa se define aleatoriamente a través de la técnica *random grid search*.

En esta investigación, diversas variantes de RNNs fueron evaluadas, por lo tanto, la arquitectura propuesta incluye una *LSTM layer*, *GRU layer*, *BiLSTM layer* y *BiGRU layer*. En otras palabras, las características de calidad se evaluaron a través de la ejecución de distintos modelos neuronales que consideran individualmente cada una de las variantes recurrentes. Con ello, se obtuvieron 28 (veintiocho) modelos neuronales que surgen al evaluar 7 (siete) características de calidad y 4 (cuatro) variantes de RNNs recurrentes (sólo si se consideran las mejores configuraciones por variante). Este tipo de redes neuronales es capaz de gestionar el procesamiento de la información secuencial a través de sus puertas. En cuanto a sus unidades ocultas (*hidden units*), también se establecieron aleatoriamente. El propósito de inclusión de más de una variante de RNNs es identificar aquella que retorna mejor resultado y con ello determinar, a través de un análisis comparativo, la más apropiada para el estudio de la calidad de los requerimientos.

También, se añadió una *Fully Connected Layer* con dimensión 2 (dos) que corresponde al número de clases, las que permiten indicar el cumplimiento de la característica de calidad bajo análisis. Luego, se incluyó una *Softmax Layer* con una función de activación sigmoide y finalmente, una *Classification Layer* que genera la salida de la red neuronal. La Figura 3.3 ilustra la arquitectura de red neuronal propuesta.

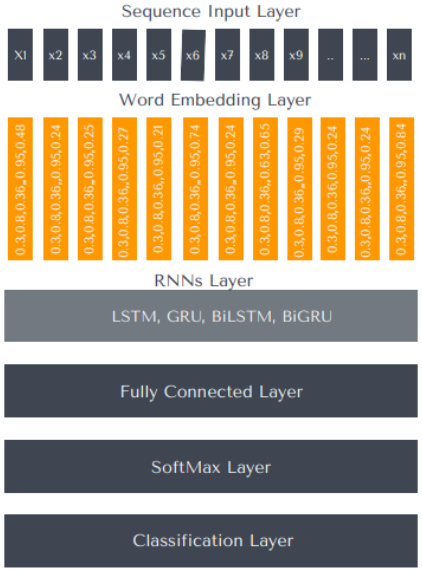


Figura 3.3: Arquitectura de red neuronal recurrente.

Por último, cada modelo ha sido definido con un enfoque de aprendizaje supervisado, puesto que los algoritmos son entrenados con datos previamente etiquetados. Finalmente, en relación a los experimentos realizados, se utilizó el 80% de la muestra para entrenar y el 20% para prueba aplicando la técnica de validación cruzada (*cross validation*). Los hiperparámetros se determinaron con una búsqueda aleatoria de valores posibles (*random grid search*). Esta técnica permite ejecutar cada modelo neuronal con un conjunto de configuraciones aleatorias, lo que facilita posteriormente identificar aquella configuración que retorna mejor rendimiento. La Tabla 3.6 sintetiza los hiperparámetros que permiten establecer las configuraciones aleatorias.

Tabla 3.6: Hiperparámetros.

Hiperparámetro	Rango
Embedding Dimensions	[32, 64, 128]
Dropout	[0.1, 0.2, 0.5]
Hidden Units	[32, 64, 128]
Mini Batch Size	16
Optimizador	Adam
Error Criteria	Binary Cross Entropy
Gradiente Threshold	2
Learning Rate	0.001
Epochs	30

3.2.7/ Fase de Evaluación

En la *fase de evaluación* los modelos de salida se evalúan sobre los conjuntos de datos de prueba utilizando métricas predefinidas. En la siguiente sección se evalúa el rendimiento de los modelos de clasificación. Luego, se provee un análisis comparativo de los resultados obtenidos en función a cada una de las variantes recurrentes (LSTM, GRU, BiLSTM y BiGRU) incluidas en esta investigación.

3.3/ Medidas de desempeño

Las métricas se utilizan para determinar el desempeño de los modelos neuronales. Particularmente, en este trabajo de Tesis se han considerado las siguientes métricas: *Recall*, *Precision*, *Accuracy* y *F-1 score*.

3.3.1/ Recall

Esta métrica, también denominada sensitividad, es la proporción de casos positivos clasificados correctamente con respecto a todos los casos reales que son positivos. Se calcula dividiendo el número de verdaderos positivos (VP) entre la suma de los verdaderos positivos y los falsos negativos (FN). El recall se centra en la capacidad del modelo para encontrar todos los casos positivos existentes. Matemáticamente, se expresa como:

$$Recall = \frac{VP}{VP + FN} \tag{3.1}$$

3.3.2/ Precision

Esta métrica, también conocida como confidencia, denota la proporción de casos positivos predichos correctamente con respecto a todos los casos clasificados como positivos. Se calcula dividiendo el número de verdaderos positivos (VP) entre la suma de los verdaderos positivos y los falsos positivos (FP). La precisión se centra en la exactitud de las predicciones positivas. Matemáticamente, se expresa como:

$$Precision = \frac{VP}{VP + FP} \tag{3.2}$$

3.3.3/ Accuracy

La exactitud representa la fracción de instancias clasificadas correctamente por el modelo, en relación con el total de instancias evaluadas. Esta métrica toma en cuenta los promedios de la *Precision* y su inversa, así como el promedio del *Recall* y su inversa. Matemáticamente, se puede expresar como:

$$Accuracy = \frac{VP + VN}{VP + FP + VN + FN} \tag{3.3}$$

3.3.4/ F1-score

Esta métrica corresponde al promedio armónico entre la *Precision* y el *Recall*. Al igual que el *Accuracy*, esta métrica considera los FP y FN. Matemáticamente, puede ser expresado como:

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{3.4}$$

3.3.5/ Matriz de confusión

La matriz de confusión proporciona los valores necesarios para calcular las métricas anteriormente descritas y, con ello, evaluar el desempeño del modelo en términos de su capacidad para clasificar correctamente las instancias positivas y negativas. Es una tabla que muestra la frecuencia con la que las instancias del conjunto de datos se clasifican correctamente o incorrectamente por cada modelo (ver Tabla 3.7). Esta tabla tiene cuatro elementos principales:

- Verdaderos positivos (VP): Representa la cantidad de instancias positivas que el modelo clasificó correctamente.
- Falsos positivos (FP): Representa la cantidad de instancias negativas que el modelo clasificó incorrectamente como positivas.
- Verdaderos negativos (VN): Representa la cantidad de instancias negativas que el modelo clasificó correctamente.

- Falsos negativos (FN): Representa la cantidad de instancias positivas que el modelo clasificó incorrectamente como negativas.

A partir de los valores contenidos en la matriz de confusión, se han calculado las métricas: *Recall*, *Precision*, *F1-score* y *Accuracy*, cuyos resultados se analizan en la siguiente sección.

Tabla 3.7: Matriz de confusión.

Valores Predichos / Reales	R+	R-
P	VP	FP
N	FN	VN

3.3.6/ Resultados y Discusiones

En primer lugar, se analizaron las configuraciones de los modelos neuronales resultantes que retornan mayor precisión. La Tabla 3.8 sintetiza las mejores configuraciones obtenidas en base a características de calidad y variante recurrente. El análisis incluye la variante más conveniente, así como los valores correspondientes a *embedding dimension*, *hidden units* y *dropout* puesto que, como se menciona en la sección anterior, los hiperparámetros han sido establecidos aleatoriamente con la técnica *random grid search* (Anexo II).

Tabla 3.8: Configuraciones resultantes por característica de calidad.

Característica Calidad	**Variante**	**Embedding Dimension**	**Hidden Units**	**Dropout**
No Ambiguo	BiLSTM	32	32	0.2
Verificable	LSTM	64	32	0
Correcto	BiLSTM	64	32	0
Completo	LSTM	128	128	0.2
Factible	LSTM	32	64	0.1
Conforme	BiLSTM	128	64	0.1
Singular	BiGRU	128	32	0

Las configuraciones resultantes muestran una tendencia favorable respecto al rendimiento de las redes neuronales recurrentes LSTM y BiLSTM. En cuanto a las BiGRU, su uso sólo resultó conveniente en la evaluación de la característica de calidad *singular*. De hecho, es destacable mencionar que el rendimiento asociado a BiGRU ha superado a los resultados (79% precisión) reportados inicialmente en Gramajo et al. (2020b). En esta investigación, el análisis estaba circunscripto sólo al estudio comparativo de LSTM versus GRU para la evaluación de la característica de calidad *singular*.

Los resultados muestran que, en promedio, los modelos de clasificación propuestos pueden alcanzar niveles de precisión de aproximadamente 75% por cada característica de calidad. La tabla 3.9 sintetiza los mejores resultados obtenidos por cada variante de RNNs considerando como medidas de desempeño *Accuracy*, *Precision*, *Recall* y *F1-score*. En este sentido, además se observa que el valor de recall correspondiente a la característica de calidad *singular* es moderadamente bajo en relación al valor de precisión. Posiblemente, el motivo esté asociado a la distribución desbalanceada de clases (Singular / No Singular) que presenta la variable en el corpus de requerimientos.

Tabla 3.9: Resultados con redes neuronales recurrentes.

Característica Calidad	Variante	Accuracy	Precision	Recall	F-1 score
Correcto	BiLSTM	0.7389	0.7292	0.7260	0.7276
Completo	LSTM	0.7222	0.7250	0.7228	0.7239
Factible	LSTM	0.7333	0.7312	0.7312	0.7312
Singular	BiGRU	0.8111	0.7087	0.6207	0.6618
No Ambiguo	BiLSTM	0.7611	0.7336	0.7354	0.7345
Verificable	LSTM	0.7500	0.7483	0.7503	0.7493
Conforme	BiLSTM	0.7500	0.7530	0.7526	0.7528

3.4/ Experimento 2

En esta sección se describe otra arquitectura para el estudio de la calidad de los requerimientos, en este caso, basada en redes neuronales transformadoras. Específicamente, se incluye Bidirectional Encoder Representations from Transformers (BERT por sus siglas en inglés), un modelo de representación del lenguaje detallado previamente en la subsección 3.2.1.2.

3.4.1/ Modelos de Clasificación basados en Redes Neuronales Transformadoras

En primer lugar, se realizó el preprocesamiento de los requerimientos. Para ello, se ha utilizado una estrategia diferente a la planteada en la propuesta anterior. Esto se debe a que en este segundo experimento se emplea BERT, un modelo neuronal de aprendizaje profundo de propósito general que ha sido previamente entrenado con un corpus distinto al propuesto originalmente en este trabajo de Tesis. BERT ha sido entrenado en un conjunto de datos masivo y diverso compuesto por una amplia variedad de textos extraídos de la web. Este conjunto de datos se diseñó para representar una muestra representativa del lenguaje humano en múltiples dominios y contextos. Aunque el tamaño exacto del conjunto de datos no se ha divulgado de manera específica, se sabe que contiene miles de millones de palabras.

El conjunto de datos utilizado para entrenar BERT incluye textos de noticias, libros, artículos en línea, sitios web, foros, redes sociales y otras fuentes disponibles públicamente en la web. Esta diversidad en las fuentes y temas permitió que BERT adquiriera un profundo entendimiento del lenguaje natural en su conjunto, en lugar de estar especializado en un dominio o contexto particular. Como resultado, BERT es capaz de capturar relaciones semánticas y contextuales en el lenguaje, lo que lo hace altamente adaptable y efectivo en una amplia gama de tareas de procesamiento de lenguaje natural (Kenton y Toutanova, 2019). Este modelo neuronal permite la generación de incrustaciones de palabras (*word embedding*) profundamente contextualizadas. Esto es posible a partir del modelado de lenguaje enmascarado y los mecanismos de atención que ofrecen las redes neuronales transformadoras utilizadas en su entrenamiento.

Para utilizar BERT en tareas específicas, como el procesamiento de requerimientos de software se llevó a cabo el entrenamiento del modelo con el conjunto de datos ultilizado en el Experimento 1. En primer lugar, es necesario convertir los datos de entrada a un formato conveniente, de manera tal que cada requerimiento pueda ingresar al modelo y, con ello, obtener las incrustaciones de palabras

(*word embeddings*) correspondientes. Por lo tanto, primero los requerimientos (*words input sequence*) se tokenizaron con el propósito de estructurarlos en unidades más pequeñas (tokens) para ello se ha utilizado BERT tokenizer. Posteriormente, los tokens que representaban palabras en mayúsculas se transformaron a minúsculas (*lowercase*).

Luego, a cada secuencia de entrada representada a nivel de token, se le añadieron las etiquetas [CLS] y [SEP]. Éstas permiten, permiten señalar el comienzo y el final de cada oración respectivamente. A continuación, las representaciones de requerimientos a nivel de tokens se transformaron en secuencias de índices escalares mediante identificadores únicos (IDs), lo que da lugar a los *segment IDs*. Estos últimos tienen correspondencia con los IDs, previamente asignados a los tokens en el entrenamiento del modelo BERT. Posteriormente, las secuencias de IDs se codificaron nuevamente en tensores binarios con las máscaras de atención (*input attention mask*) que luego, en el entrenamiento, permiten indicar al modelo BERT cuáles tokens se deben atender primero.

Finalmente, la longitud (L) para cada secuencia de entrada, es decir, por cada requerimiento, se estableció en 100 palabras. Esto es así, dado que se considera la L máxima disponible en el corpus. Si los requerimientos poseen una longitud menor a L, se adopta la estrategia *pre-sequence padding* (Anexo.1). La Figura 3.4 ilustra el preprocesamiento realizado.

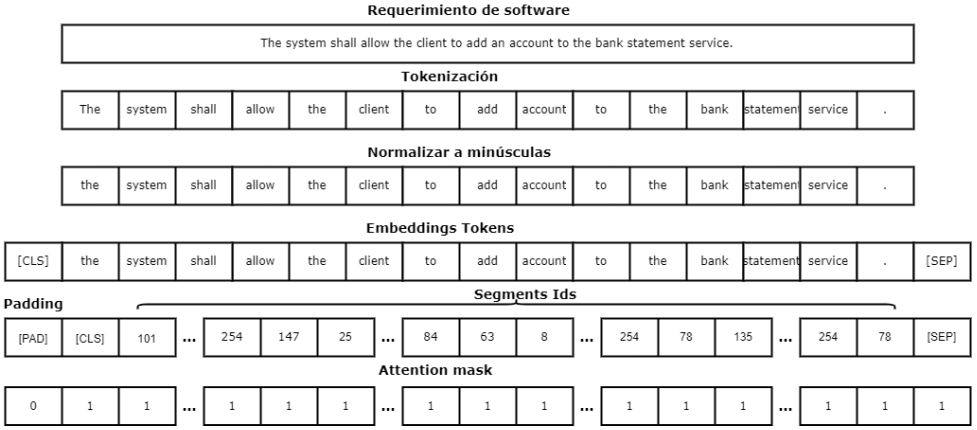


Figura 3.4: Preprocesamiento de requerimientos.

Por otro lado, la arquitectura propuesta que permite generar modelos neuronales profundos para evaluar cada característica de calidad incluye una *Input Layer* compuesta por *words input sequence*, *segment IDs* y, por último, *input mask*. Luego, se añadió una BERT Layer que permite obtener las incrustaciones de palabras (*words embeddings*) correspondientes. Posteriormente, se realizó el ajuste fino (*fine-tuned*), es decir, se definió el propósito de la arquitectura neuronal. Particularmente, se pretendía construir un clasificador binario por cada característica de calidad, por lo que se añaden dos *Fully Connected Layer*. Por último, se incluyó una *Softmax Layer* con una función de activación sigmoide que genera la salida de la red neuronal. La Figura 3.5 ilustra la arquitectura propuesta.

Del mismo modo que en el experimento anterior, cada modelo sigue un enfoque de aprendizaje supervisado, dado que los algoritmos son entrenados con datos previamente etiquetados. El 80% de la muestra se utilizó para entrenar y el 20% para prueba aplicando la técnica de validación cruzada (*cross validation*). Los hiperparámetros se determinaron con una búsqueda aleatoria de valores posibles (*random grid search*). La Tabla 3.10 sintetiza los hiperparámetros.

Figura 3.5: Arquitectura basada en redes neuronales transformadoras.

3.5/ Resultados y Discusiones

En esta sección se analizan las configuraciones de los modelos neuronales resultantes que retornan mayor precisión. La Tabla 3.11 sintetiza las mejores configuraciones obtenidas por cada característica de calidad. Esta última incluye los valores correspondientes a las *Hidden Units* para cada *Fully Connected Layer* añadida, así como, los valores de *Dropout* puesto que, como se menciona en la sección anterior, los hiperparámetros se establecen aleatoriamente con la técnica *random grid search* (Anexo II).

Tabla 3.10: Hiperparámetros.

Hiperparámetro	**Rango**
Dropout	[0.1, 0.2]
Hidden Units	[32, 64, 128]
Batch Size	16
Optimizador	Adam
Error Criteria	Categorical Cross Entropy
Gradiente Threshold	5
Learning Rate	0.001
Epochs	30

Principalmente, se observa que la inclusión de BERT permite obtener resultados con una mejora sustancial en la precisión de la *característica de calidad No Ambiguo*. En contraste con el resto de las características de calidad, cuyo rendimiento es inferior al obtenido aplicando RNNs. La Tabla 3.11 provee una síntesis de los mejores resultados obtenidos considerando como medidas de desempeño *Accuracy*, *Precision*, *Recall* y *F1-score*.

Desafortunadamente, los resultados muestran que, en promedio, los modelos de clasificación propuestos pueden alcanzar niveles de precisión de aproximadamente el 65%. En efecto, las configuraciones resultantes muestran una tendencia favorable respecto al rendimiento de las RNNs, en contraste a aquellas basadas en redes neuronales transformadoras. En el Anexo II se pueden visualizar las ejecuciones

Tabla 3.11: Resultados con BERT.

Característica Calidad	Accuracy	Precision	Recall	F-1 score
No Ambiguo	0.778	0.7750	0.7750	0.7750
Singular	0.7847	0.7937	0.7937	0.7937
Correcto	0.6667	0.6562	0.6562	0.6562
Factible	0.5417	0.5312	0.5312	0.5312
Conforme	0.7431	0.7437	0.7437	0.7437
Verificable	0.6736	0.6750	0.6750	0.6750
Completo	0.5278	0.5188	0.5188	0.5187

de los modelos neuronales con las configuraciones que retornan mayor precisión.

3.5.1/ Características de Software y Hardware

Por último, en esta sección se mencionan las características de software y hardware utilizadas en la ejecución de los modelos neuronales que se sintetizan en la Tabla 3.12.

Tabla 3.12: Características de software y hardware.

Recurrentes	Transformadoras
NVIDIA Geforce GTX 1660Ti	NVIDIA-SMI 470.82.01
CUDA Version: 11.8	CUDA Version:11.4
Memory 40GB DDR4	Memory 13GB DDR4
Windows 10	Linux 6 5.15 SMP
CPU AMD Ryzen 5 3600X 3.80 GHz	CPU Intel(R) Xeon(R) 2.00GHz
Matlab Version: 9.10	Python Version: 3.7.6.

En el siguiente, Capítulo 4, se describe un framework desarrollado para la evaluación automática de las características de calidad. Por lo tanto, se describen allí las fases de implementación de los modelos neuronales y su correspondiente monitoreo.

3.5.2/ Amenazas a la Validez

En esta sección se discuten las posibles amenazas a la validez respecto a la propuesta para la evaluación de calidad.

Dado que los modelos neuronales tienen un enfoque de aprendizaje del tipo supervisado, se identifican potenciales amenazas que podrían afectar su validez. La razón principal es que el rendimiento de los modelos definidos es altamente dependiente de sus entradas, es decir, de los requerimientos previamente etiquetados por los expertos. A pesar de los esfuerzos para disminuir los sesgos en este aspecto, los requerimientos etiquetados que se utilizan durante el entrenamiento de los modelos podrían representar una posible amenaza. Por último, cabe mencionar que durante el proceso de etiquetado de datos se ha considerado la guía propuesta por INCOSE (2017) para mitigar posibles errores en los valores de

entrada. Dicho documento proporciona algunas pautas que ayudan a los ingenieros a determinar el cumplimiento de las características de calidad de forma manual.

3.6/ Conclusiones

En este capítulo, se realiza un análisis exhaustivo de las dificultades y limitaciones identificadas en propuestas anteriores, relacionadas con la representación del contexto y el conjunto de datos reducido disponible para el entrenamiento de los algoritmos de aprendizaje automático. Estos desafíos son abordados en la presente investigación integrando diferentes conjuntos de datos en un corpus de requerimientos enriquecido. Posteriormente, se provee una descripción detallada de las fases que componen la metodología utilizada para construir los modelos neuronales. Además, se realiza un análisis detallado de cada una de las características de calidad, evaluando su viabilidad para ser abordadas mediante técnicas de aprendizaje profundo. Se examinan aspectos tales como la disponibilidad de datos relevantes y etiquetados para entrenar y evaluar los modelos, así como la capacidad de las redes neuronales para capturar y modelar las características y patrones asociados con cada propiedad de calidad.

Luego, se presentan dos potentes arquitecturas neuronales de aprendizaje profundo: *Redes Neuronales Recurrentes* y *Redes Neuronales Transformadoras* para abordar la evaluación de la calidad de los requerimientos. Diversas variantes de redes neuronales recurrentes fueron analizadas: LSTM, GRU, BiLSTM y BiGRU. También se evaluó el rendimiento de BERT, un modelo de aprendizaje profundo previamente entrenado basado en redes neuronales transformadoras. Cada una de estas arquitecturas se examina en términos de su capacidad para capturar y modelar las dependencias temporales presentes en los requerimientos, lo que resulta fundamental para la evaluación de la calidad. Por lo que se provee un estudio comparativo respecto a los resultados obtenidos. Estos últimos han demostrado que las redes neuronales recurrentes presentan un rendimiento superior a las obtenidas con BERT, en este contexto específico.

3.7/ Summary

In this chapter, a comprehensive analysis is conducted on the difficulties and limitations identified in previous proposals related to context representation and the limited dataset available for training machine learning algorithms. These challenges are addressed in the present research by integrating different datasets into an enriched corpus of requirements. Subsequently, a detailed description is provided of the phases that compose the methodology used to build the neural models. Additionally, a thorough analysis is conducted on each of the quality characteristics, evaluating their viability to be addressed through deep learning techniques. Aspects such as the availability of relevant and labeled data for training and evaluating the models are examined, as well as the ability of neural networks to capture and model the features and patterns associated with each quality property.

Next, two powerful architectures of deep learning neural networks are presented: Recurrent Neural Networks (RNNs) and Transformer Neural Networks, to address the evaluation of requirement quality. Various variants of recurrent neural networks are analyzed, including LSTM, GRU, BiLSTM, and BiGRU. The performance of BERT, a pre-trained deep learning model based on Transformer neural networks, is also evaluated. Each of these architectures is examined in terms of their ability to capture and model the temporal dependencies present in the requirements, which is crucial for quality evaluation. A

comparative study is provided regarding the results obtained. These findings have shown that recurrent neural networks outperform the performance achieved with BERT in this specific context.

4

Framework para la Evaluación Automática de la Calidad de Requerimientos

Este capítulo describe el desarrollo de una herramienta que integra en un entorno web los modelos neuronales generados. El propósito es evaluar de forma automática la calidad de los requerimientos. En primer lugar, se describen las características principales de la herramienta de soporte a las tareas de verificación de requerimientos. Luego, se presentan artefactos de diseño que cubren el modelado del sistema, considerando las notaciones de Lenguaje de Modelado Unificado. Posteriormente, se realiza un análisis comparativo respecto a otras herramientas presentes en la literatura. Por último, se analiza la integración de la herramienta en el proceso de la Ingeniería de Requerimientos.

4.1/ Introducción

Este trabajo de Tesis propone el desarrollo de una herramienta que integra tecnologías de IA, a fin de proveer mecanismos para el control y evaluación automática de la calidad de requerimientos expresados en lenguaje natural. El propósito principal de la herramienta es brindar soporte a los profesionales del software en la toma de decisiones, sin demandar esfuerzos y trabajo adicional. Del mismo modo, facilitar la ejecución de tareas que refieren a la verificación de requerimientos.

En resumen, este trabajo de Tesis no se limita sólo a la investigación sobre aplicaciones de redes neuronales para el estudio de la calidad de los requerimientos, sino que además integra los resultados obtenidos en una herramienta puesta a disposición de los profesionales del software para su uso en el proceso de IR. Por consiguiente, esta propuesta permite a la IR beneficiarse de las tecnologías de IA y con ello, definir un flujo de trabajo inteligente que conduzca al desarrollo de productos de software de calidad. En las siguientes secciones se introducen las principales características de la herramienta que se desarrollan en el marco de las fases de implementación y monitoreo correspondientes a la metodología de trabajo descrita en el Capítulo 3.

4.1.1/ Fases de Implementación y Monitoreo

Las fases de implementación y monitoreo comprenden el despliegue del código de inferencia de los modelos neuronales obtenidos en un dispositivo o entorno y éstos se supervisan de manera continua para detectar posibles errores durante su ejecución. En este contexto, se desarrolla una herramienta web para integrar los modelos neuronales obtenidos y, con ello, proveer soporte a las tareas de verificación de requerimientos.

4.1.2/ Herramienta para Evaluar la Calidad de Requerimientos

4.1.2.1/ Quality Assessment Support Tool

En esta sección se presenta *Quality Assessment Support* (QASS) una herramienta que integra sólo aquellos modelos neuronales obtenidos que proveen mayor precisión según cada característica de calidad, a través de las fases descritas en el Capítulo 3, como servicios de IA en un entorno web. El objetivo de la herramienta es automatizar la evaluación de la calidad de los requerimientos. Particularmente, en este primer módulo se lleva a cabo el despliegue de los modelos neuronales embebidos y previamente entrenados que permiten la evaluación de las características de calidad desde una perspectiva individual, detallada en el Capítulo 2. Por lo tanto, se aborda la evaluación de las características de calidad: *Singular*, *No Ambiguo*, *Factible*, *Completo*, *Correcto*, *Conforme* y *Verificable*.

La herramienta está integrada en un marco de trabajo modular que proporciona múltiples servicios para llevar a cabo las actividades de validación de requerimientos en lenguaje natural. Un aspecto clave de la herramienta es que su uso no requiere experiencia previa en aprendizaje automático. Entre sus principales características se destacan:

- Carga de requerimientos de software en lotes.
- Base de datos integrada para el almacenamiento de requerimientos.
- Alta, baja y modificación de requerimientos.
- Integración de servicios de IA basados en redes neuronales profundas para la evaluación automática de la calidad.
- Generación de reportes que favorecen la visualización e interpretación de resultados.

A partir de esta herramienta se pretenden agilizar los flujos de trabajo propios de la IR, automatizando tareas de control y verificación de calidad y, con ello, minimizar el uso de los recursos asociados a tiempo y costos del proyecto de software. Por último, para su correspondiente inclusión en el proceso de IR, se define un flujo de trabajo el cual se describe en la siguiente sección.

4.1.3/ Flujo de Trabajo

Con el propósito de facilitar las tareas de verificación a través del uso de la herramienta QASS, se propone un flujo de trabajo compuesto por 4 (cuatro) pasos:

- *Ingresar Requerimientos* provee los mecanismos necesarios para la carga de los requerimientos que serán sometidos a evaluación. El proceso de carga admite el ingreso de requerimientos individuales o de un conjunto de requerimientos.
- *Ajustar Requerimientos* permite añadir, editar y eliminar requerimientos. Este paso es opcional y provee los mecanismos para realizar ajustes convenientes sobre los requerimientos cargados previamente a su evaluación.
- *Evaluar Requerimientos* es posible ejecutar los modelos neuronales integrados a la herramienta web con una configuración predeterminada, es decir, con aquellas combinaciones de hiperparámetros que proveen mayor rendimiento, resultantes de las fases previas de entrenamiento y evaluación descritas en el Capítulo 3. En otras palabras, no se requiere que el Ingeniero de Requerimientos establezca los valores de hiperparámetros o cuestiones relacionadas a la arquitectura de las redes neuronales, tales como cantidad de capas, funciones de activación, variantes de redes neuronales, entre otras, para obtener las predicciones provistas por los modelos. Sin embargo, no se desestima escalar las funcionalidades de la herramienta añadiendo opciones para los usuarios más avanzados con conocimientos de aprendizaje automático, de forma tal de proveer opciones de ajustes sobre hiperparámetros, métricas y partición del conjunto de datos.
- *Verificar Requerimientos* permite visualizar los resultados obtenidos luego de la ejecución de los modelos neuronales embebidos. Los requerimientos evaluados por la herramienta se despliegan en formato tabular y el cumplimiento de las características de calidad se indica mediante variables dicotómicas. Esto permite a los ingenieros determinar la calidad de los requerimientos de forma rápida y sencilla. Luego de monitorear los resultados provistos por la herramienta, el Ingeniero de Requerimientos es capaz de realizar correcciones pertinentes sobre los requerimientos evaluados de forma automática, a través de los mecanismos de edición si esto último fuera necesario. Es decir, que es posible editar aquellos valores dicotómicos retornados por la herramienta si el ingeniero lo considera pertinente.

Finalmente, un aspecto clave que se destaca, es que la herramienta desarrollada constituye una tecnología de soporte a las tareas de verificación de requerimientos y no de reemplazo al profesional del software.

4.2/ Modelado de la Herramienta QASS

A fin de llevar a cabo el desarrollo de la herramienta de soporte a la evaluación de la calidad, en esta sección se presentan los principales artefactos de diseño (diagramas de caso de uso, diagramas de secuencia y diagramas actividad) que permiten su modelado. Particularmente, se consideran las notaciones de Lenguaje de Modelado Unificado (UML), que facilitan la visualización, descripción y documentación de la herramienta desde una perspectiva de alto nivel (Larman, 2012).

4.2.1/ Diagrama de Caso de Uso

Los diagramas de caso de uso exponen las interacciones entre el sistema y su entorno. En esta primera versión de la herramienta, se estableció como alcance la inclusión de los casos de uso que corresponden a *Cargar Lote de Requerimientos*, *Ajustar Lotes de Requerimientos* con sus casos de uso extendidos:

Añadir Requerimiento, *Editar Requerimiento*, *Eliminar Requerimiento* y, por último, *Evaluar Requerimientos* y *Verificar Requerimientos*. La Figura 4.1 ilustra el diagrama de casos de uso. Cabe destacar que en el Anexo III se incluyen las descripciones de los casos de uso.

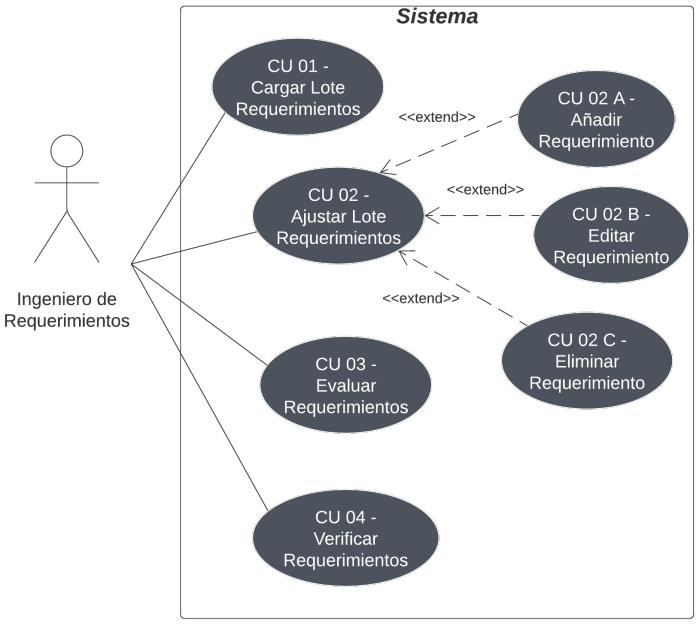


Figura 4.1: Diagrama de caso de uso.

4.2.2/ Diagrama de Actividad

Los diagramas de actividad se utilizan para modelar el flujo de las actividades incluidas en el sistema. El diagrama de actividades correspondiente a la herramienta QASS, se ilustra en la Figura 4.2.

El proceso comienza con la carga de los requerimientos, los cuales se muestran en un formato tabular para facilitar su visualización. Opcionalmente, el Ingeniero de Requerimientos puede realizar ajustes sobre los requerimientos (añadir, editar y eliminar). Estos últimos son registrados en la base de datos por el sistema. A continuación, se lleva a cabo la evaluación de los requerimientos mediante la ejecución de los servicios de IA embebidos en la herramienta. Del mismo modo, los resultados de la evaluación automática de los requerimientos se presentan en forma tabular y el cumplimiento de las características de calidad evaluadas se indican con variables dicotómicas que permiten la interpretación de los resultados de forma intuitiva. Por último, es posible verificar los resultados obtenidos y corregirlos oportunamente.

4.3/ Interfaces de Usuario

Para llevar a cabo las tareas de verificación de la calidad de los requerimientos de forma automática se definió un flujo de trabajo, descritos en la Sección 4.1.3. Dicho flujo, es puesto a disposición de los Ingenieros de Requerimientos a través de la herramienta QASS.

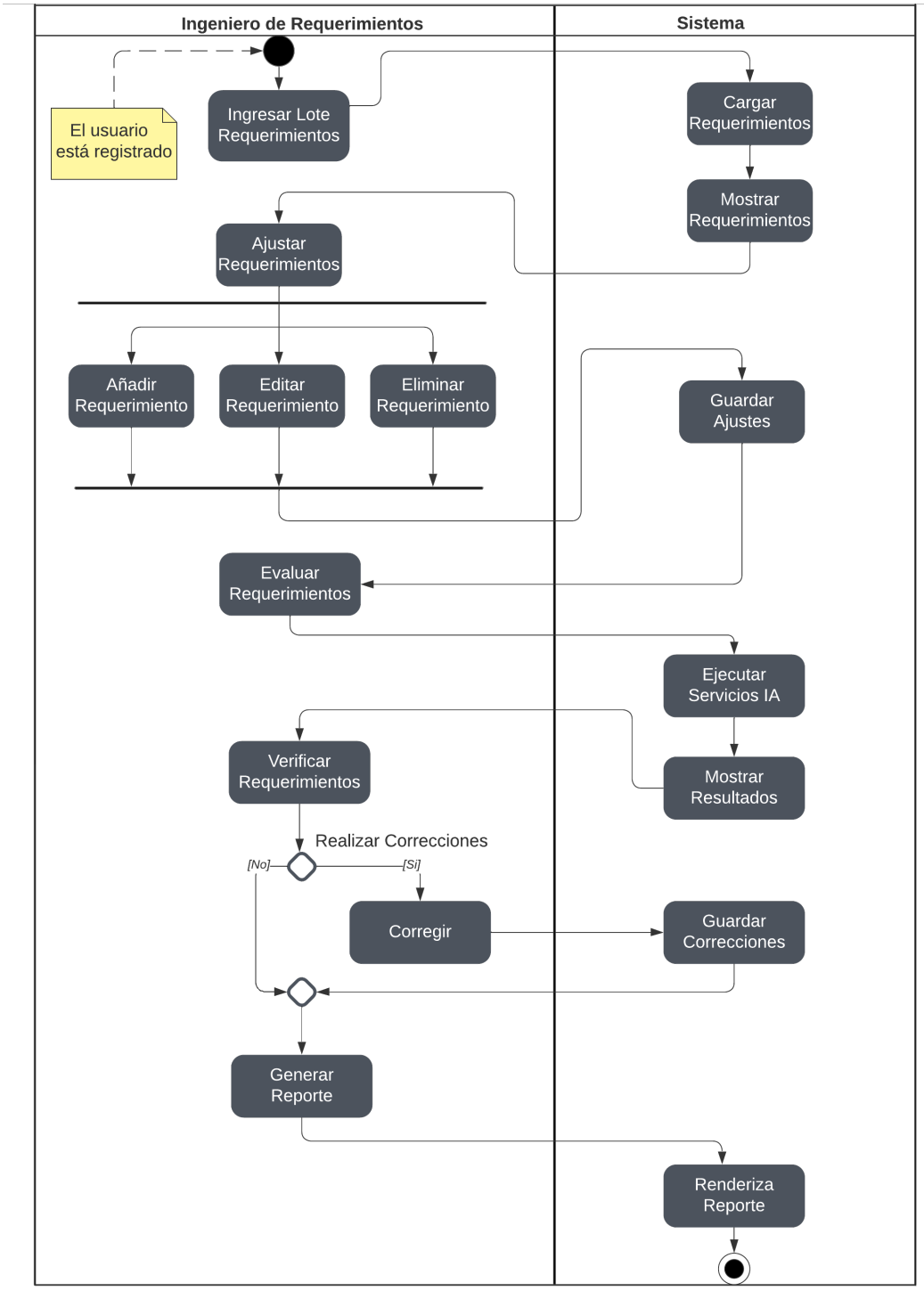


Figura 4.2: Diagrama de actividad.

Para iniciar el flujo de trabajo que conduce a la verificación de los requerimientos la herramienta provee dos alternativas. Una de ellas, *Evaluate requirements*, a través de la cual el servicio de evaluación basado en IA es ejecutado con la configuración predeterminada. En otras palabras, no admite ajustes sobre los hiperparámetros de los modelos. Por otro lado, la opción *Train your own model*

permitirá realizar ajustes sobre los valores de hiperparámetros y métricas de evaluación. Tal como se mencionó previamente, esta última forma parte del conjunto de características que serán implementadas en próximas versiones. Ésta se incluye en la interfaz inicial para ilustrar la magnitud y el alcance de la herramienta QASS. Por lo tanto, en esta primera instancia, sólo es posible iniciar el flujo de trabajo a través del botón *Evaluate requirements*. La Figura 4.3 ilustra las opciones de inicio para el flujo de trabajo definido.

Figura 4.3: Interfaz Iniciar Flujo de Trabajo.

4.3.1/ Paso 01 - Ingresar Requerimientos

El proceso de carga de requerimientos admite la inserción de requerimientos individuales o por lotes. En ambos casos los requerimientos deben estar contenidos en un archivo de texto plano. El formato de archivo admitido por la herramienta es CSV. La Figura 4.4 muestra la interfaz correspondiente a la carga de requerimientos.

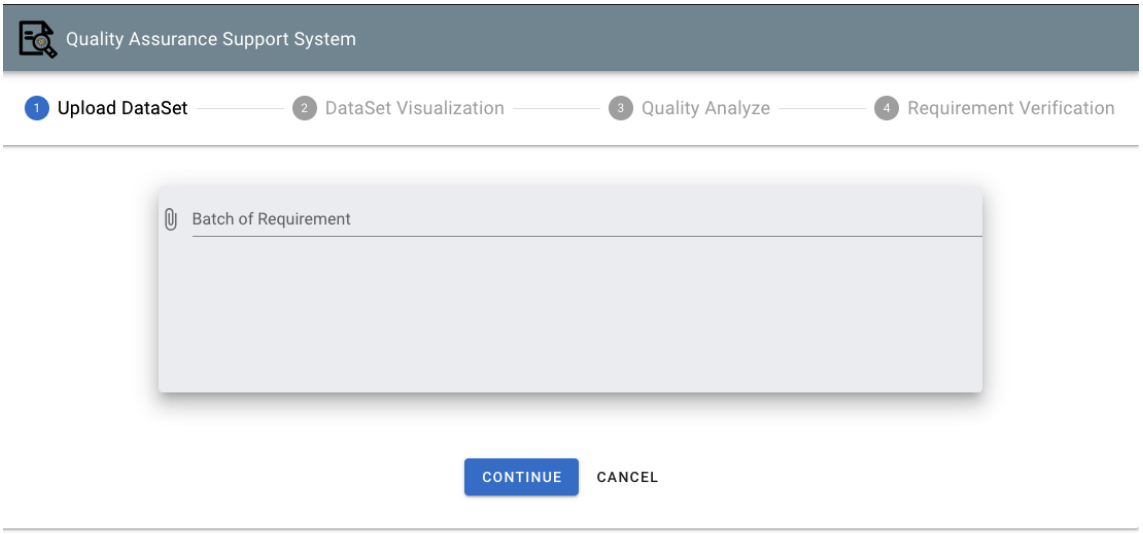


Figura 4.4: Interfaz Ingresar Requerimientos.

4.3.2/ Paso 02 - Ajustar Requerimientos

La herramienta QASS provee la capacidad de realizar opcionalmente ajustes sobre los requerimientos cargados, antes de iniciar el proceso de evaluación (4.5). Por lo tanto, el Ingeniero de Requerimientos es capaz de realizar las siguientes acciones:

- Añadir nuevo requerimiento: esta funcionalidad permite la inserción de un nuevo requerimiento mediante el botón *"New Requirement"* (Figura 4.6).
- Editar requerimiento: es posible la edición de la descripción de un requerimiento mediante el botón *"Edit"* (Figura 4.7).
- Eliminar requerimiento: la herramienta permite la eliminación de un requerimiento a través del botón *"Delete"* (Figura 4.8).

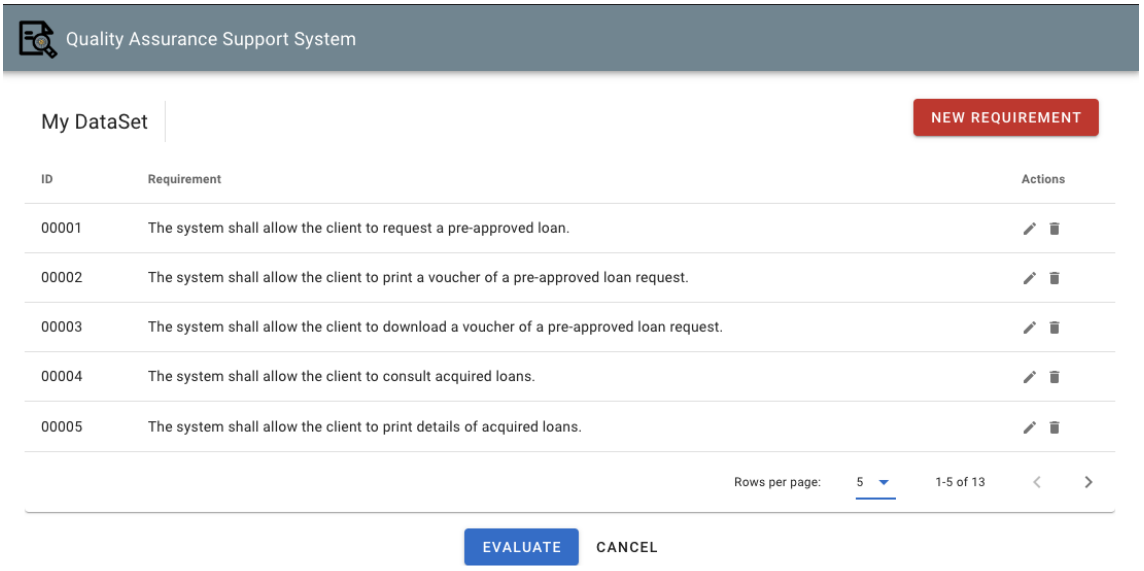


Figura 4.5: Interfaz Ajustar Requerimientos.

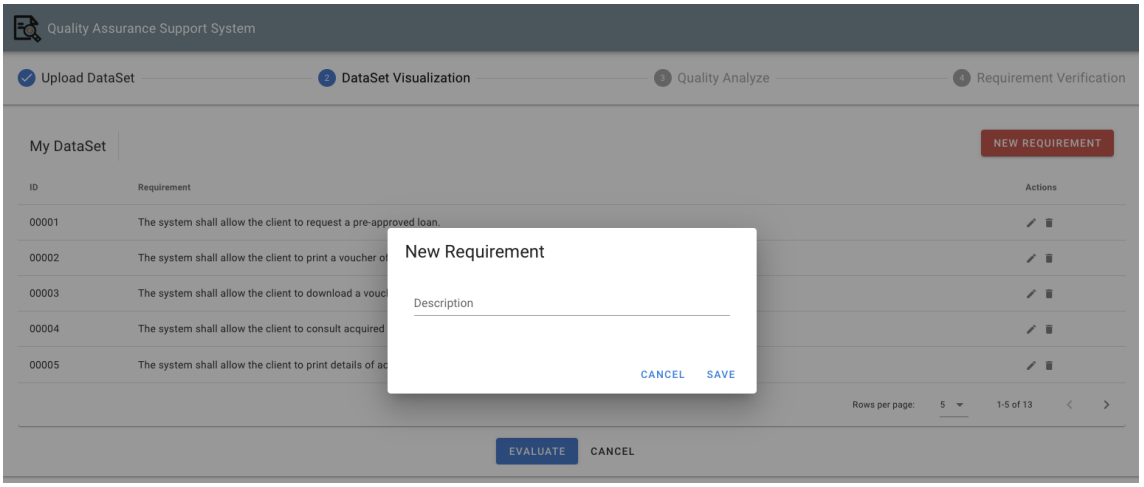


Figura 4.6: Interfaz Nuevo Requerimiento.

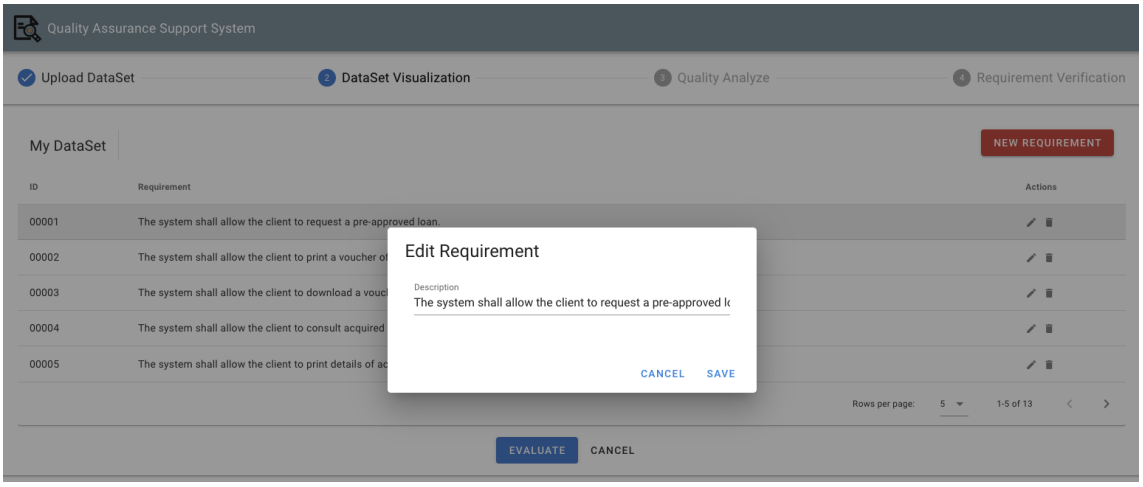


Figura 4.7: Interfaz Editar Requerimiento.

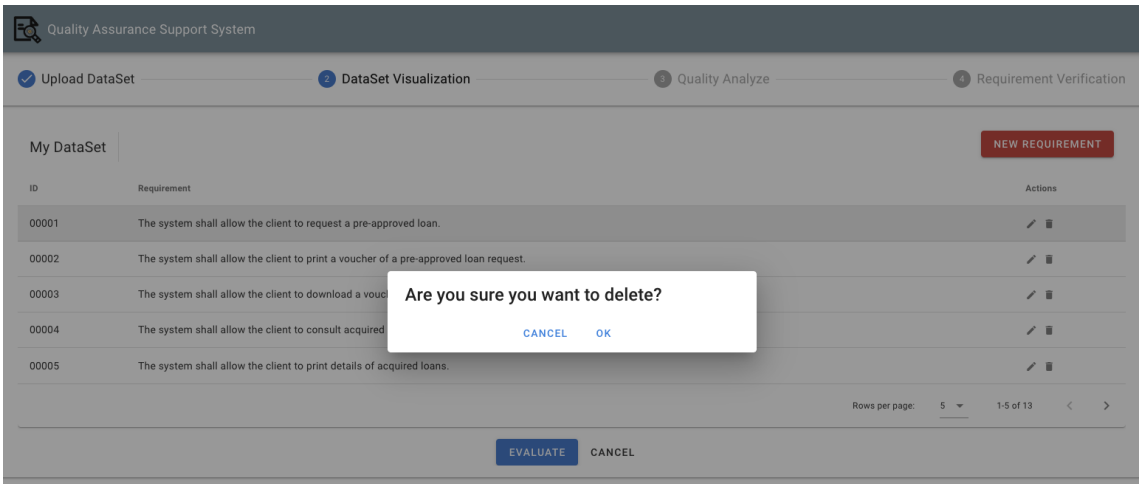


Figura 4.8: Interfaz Eliminar Requerimiento.

4.3.3/ Paso 03 - Evaluar Requerimientos

Luego, la herramienta QASS provee una vista tabular de los requerimientos cargados que serán posteriormente evaluados. En efecto, esta característica permite al Ingeniero de Requerimientos visualizar y revisar los requerimientos de forma fácil y rápida antes de la evaluación.

El Ingeniero de Requerimientos es capaz de iniciar la ejecución del servicio que permite evaluar automáticamente la calidad de los requerimientos mediante el botón *"Evaluate"*. Una vez finalizada la evaluación automática de los requerimientos, los resultados se presentan en formato tabular. Los requerimientos evaluados por QASS se identifican mediante un identificador único. Del mismo modo, la herramienta muestra su correspondiente descripción y, además, el cumplimiento de las características de calidad se indica mediante el uso de variables dicotómicas (Yes/No). Para facilitar la interpretación de los resultados, el Ingeniero de Requerimientos puede acceder a las representaciones gráficas provistas por la herramienta, las cuales sintetizan la información. Por último, la herramienta también provee los mecanismos necesarios para su impresión. La Figura 4.9 ilustra la interfaz correspondiente a Evaluar Requerimientos.

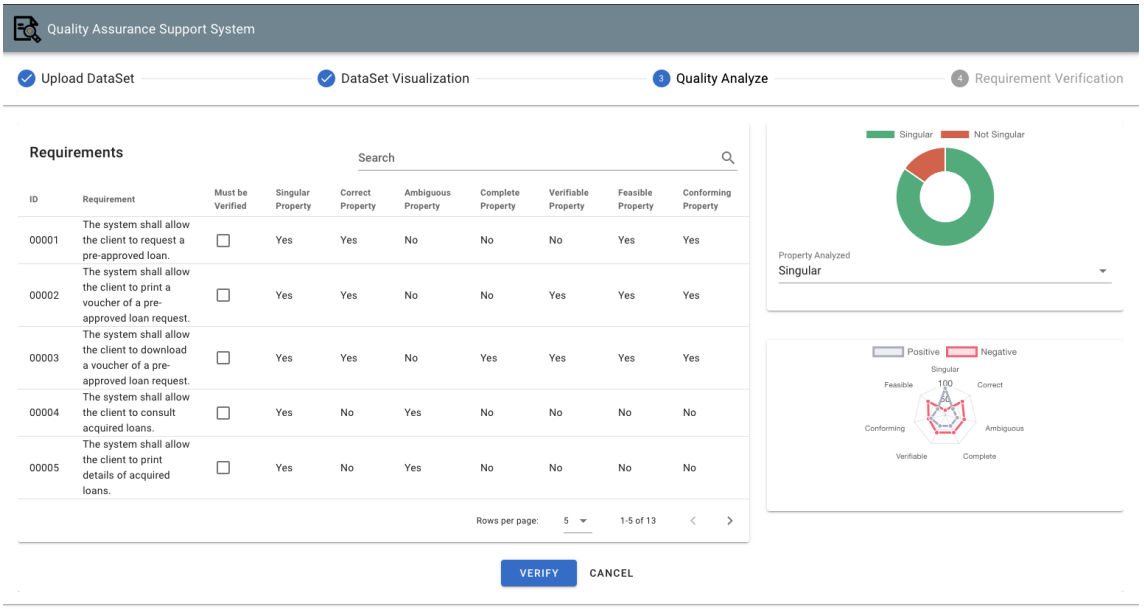


Figura 4.9: Interfaz Evaluar Requerimientos.

4.3.4/ Paso 04 - Verificar Requerimientos

El Ingeniero de Requerimientos puede verificar los resultados proporcionados por la herramienta mediante la selección de requerimientos que requieran una revisión adicional. Posteriormente, utilizando el botón *"Verify"*, la herramienta agrupa exclusivamente los requerimientos seleccionados. De esta manera, el profesional tiene la capacidad de realizar correcciones, modificando los valores dicotómicos (SI/NO) de cada característica de calidad. Esto permite ajustar los resultados obtenidos para aquellos requerimientos con los cuales el ingeniero pueda estar en desacuerdo en función de su experiencia en el área. La Figura 4.10 muestra la interfaz correspondiente a la Verificación de Requerimientos.

Una vez que se han registrado las correcciones, el sistema ofrece las funcionalidades necesarias para generar un informe en formato impreso. Por último, la herramienta QASS también proporciona la opción de revertir modificaciones incorrectas en caso de ser necesario (Botón Eliminar). La Figura 4.11 ilustra la interfaz que permite eliminar correcciones.

4.4/ Diseño Arquitectónico de Software

En el marco del desarrollo de software, la arquitectura de software representa la decisión de diseño más temprana y está determinada por los requerimientos de software. Para establecer el diseño arquitectónico es importante entender la organización y la estructura global del sistema. Este proceso consiste en identificar los componentes estructurales y la relación entre ellos. La salida del proceso de diseño arquitectónico resulta en un modelo que describe la forma en que se organiza el sistema, a través de un conjunto de componentes de comunicación (Sommerville, 2011). En esta sección se describe la arquitectura de software para la herramienta de soporte propuesta.

La arquitectura de software definida se basa en el patrón *Modelo - Vista - Controlador* (MVC). Por lo tanto, el sistema se estructura en tres componentes lógicos que interactúan mutuamente (Reenskaug, 1979). El componente *Modelo* gestiona los datos del sistema y las operaciones asociadas a ellos.

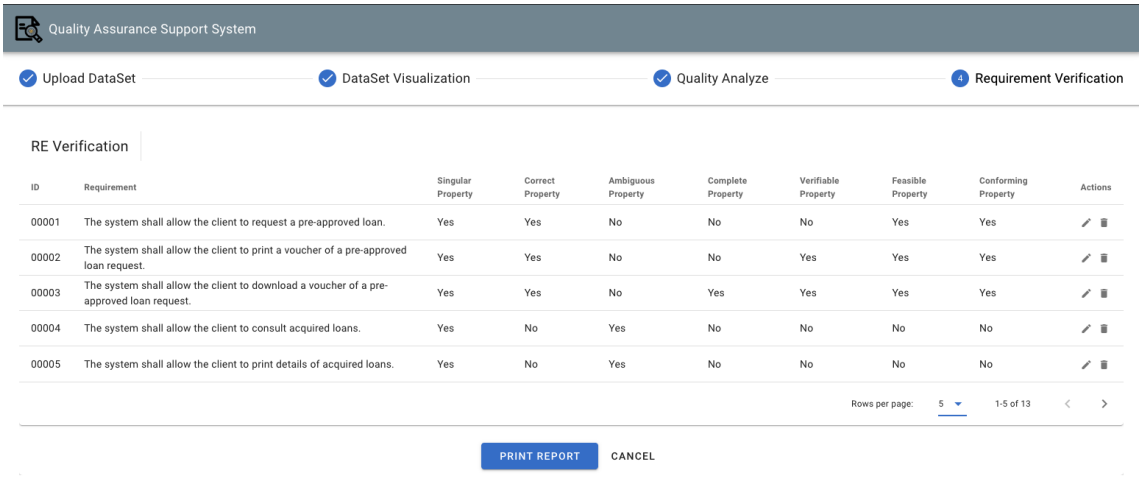


Figura 4.10: Interfaz Requerimientos para Corrección.

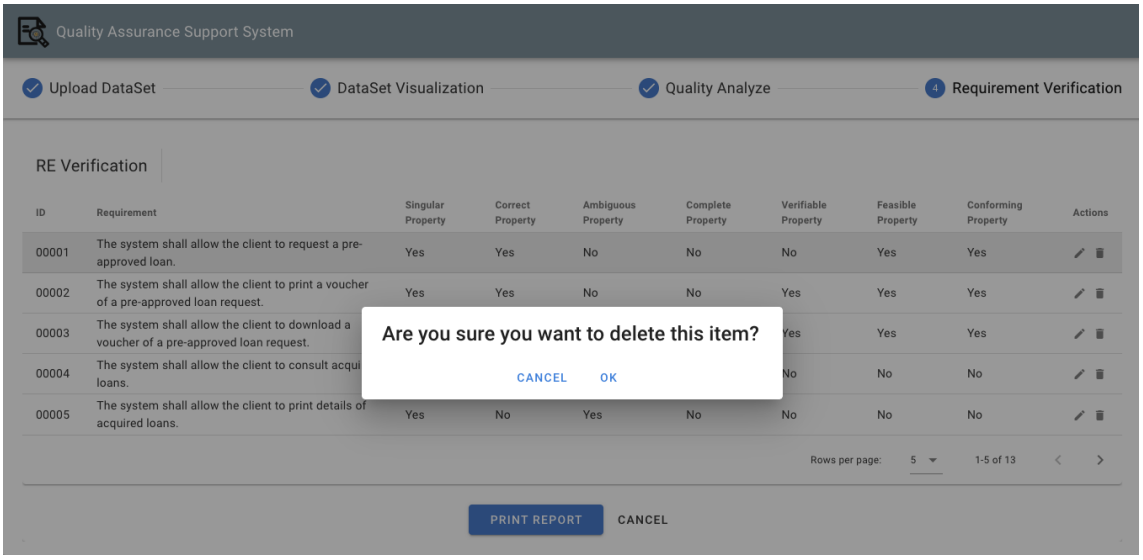


Figura 4.11: Interfaz Requerimientos para Eliminar Corrección.

También incluye la lógica de negocio y mecanismos de persistencia. Particularmente, se propone el almacenamiento de los datos que representan información asociada a los requerimientos en una base de datos relacional (Amazon RDS). Mientras tanto, los modelos neuronales estarán almacenados en contenedores en la nube (Bucket).

El componente *Controlador* actúa de intermediario entre el modelo y la vista gestionando el flujo de información entre ellos y las transformaciones necesarias para adaptar los datos a las necesidades de cada uno. En este caso, las APIs reciben y envían datos del tipo JSON provenientes del componente *Vista* y éstos son recibidos por los controladores. Luego, los controladores son capaces de realizar llamadas a los métodos que funcionan como servicios. Estos últimos realizan el procesamiento de la información accediendo a las bases de datos. Finalmente, el componente *Vista* define y gestiona cómo se presentan los datos al usuario, al igual que los mecanismos de interacción. Específicamente, se propone el uso de VUEJS y APIs para las interacciones de datos que darán lugar a la visualización de las interfaces de usuario. La Figura 4.12 ilustra la arquitectura de software correspondiente a la herramienta de soporte

que se propone en este trabajo de Tesis.

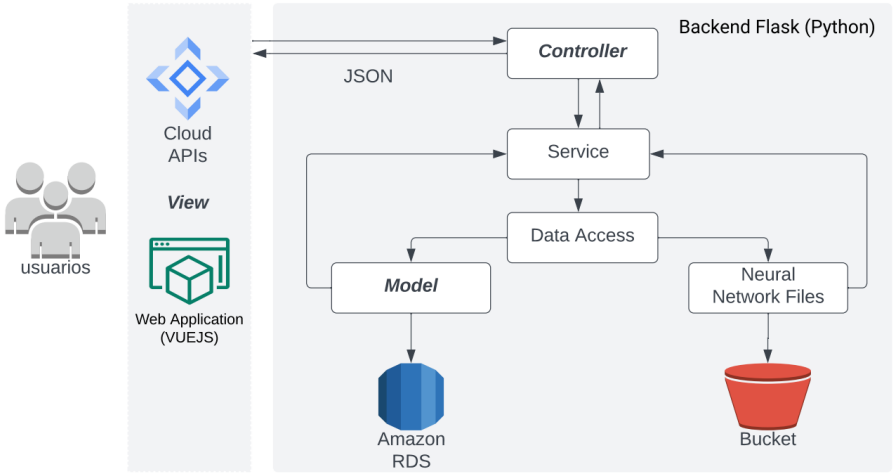


Figura 4.12: Diseño arquitectónico de software.

4.5/ Diseño de Infraestructura

Para la infraestructura de la herramienta web se propone la integración de dos servidores para Frontend y dos servidores para Backend. El Backend debe ser capaz de gestionar las conexiones con dos tipos de bases de datos: *Bucket* y *Relacional*. En lo que respecta al Frontend, éste se conecta al Backend a través de un balanceador de carga (Load Balancer). El propósito de este último es redireccionar peticiones para que los servidores no resulten saturados. Acceder a la aplicación será posible mediante una URL registrada en un servicio de hosting (Route 53). Este servicio es capaz de redirigir el tráfico de usuarios que llegan a la puerta de enlace de la aplicación (Application Gateway). Del mismo modo, también redirige el tráfico al balanceador de carga (Load Balancer) siendo éste capaz de decidir el servidor de Frontend al cual se deben enviar las peticiones de usuario. La Figura 4.13 ilustra el diseño de la infraestructura propuesta para la herramienta.

4.6/ Análisis Comparativo de Herramientas para la Evaluación de la Calidad de Requerimientos

En esta sección se provee un análisis comparativo sobre funcionalidades y características provistas por las principales herramientas de la industria del software que abordan el análisis y evaluación de la calidad de los requerimientos, a través de la aplicación de tecnologías de IA. Entre ellas se distinguen:

Requirements Quality Analyzer, es una herramienta de software que contribuye a definir, medir, gestionar y mejorar la calidad de las especificaciones de los requerimientos. Sus principales características son:

- La calidad es abordada desde un enfoque que evalúa la corrección, consistencia y completitud de los requerimientos.

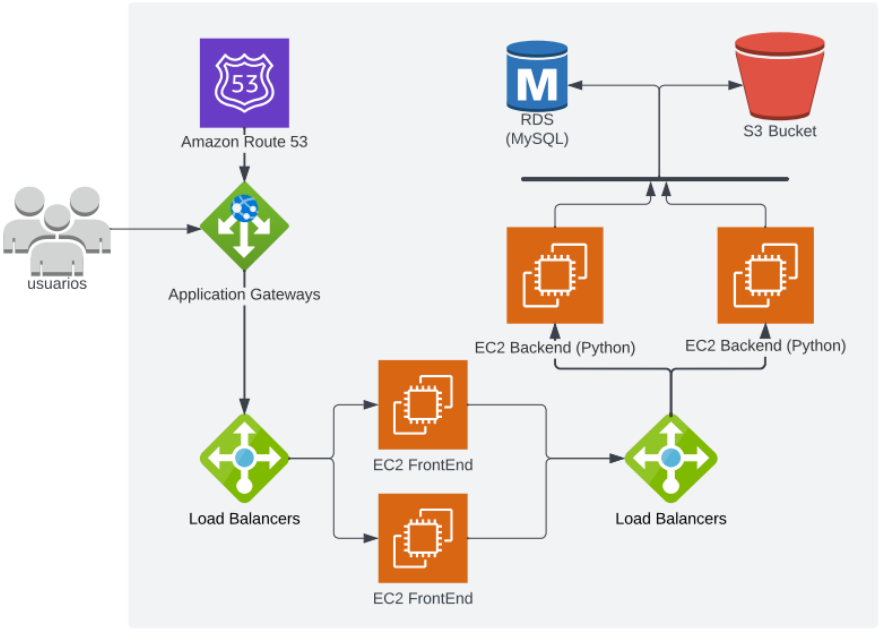


Figura 4.13: Diseño de infraestructura.

- Dispone de métricas parametrizables, métricas de código personalizado para usuarios avanzados y métricas basadas en listas de verificación que permiten una verificación manual.
- Soporte a métricas individuales (requerimiento a requerimiento) y por especificación (conjunto de requerimientos)
- Integra documentación sobre estándares y guías que refieren a la calidad para consultas en línea.

Engineering Requirements Quality Assistant es una herramienta de gestión de requerimientos diseñada para reducir retrasos en los proyectos y los costos de los equipos de ingeniería, a través de Watson IA. Por lo tanto, es capaz de inyectar inteligencia artificial en la gestión de requerimientos. Específicamente, la herramienta integra Watson Natural Language Understanding por lo que es potencialmente robusta y puede cubrir el 80% de las pautas propuestas por INCOSE (2017). Entre sus principales características se puede mencionar:

- Capacidad para detectar requerimientos compuestos, requerimientos negativos, ambigüedad, voz pasiva, requerimientos incompletos, y cantidades no específicas.
- Del mismo modo, es capaz de capturar los comentarios del Ingeniero de Requerimientos realizados sobre los documentos, por lo que, cuanto más se usa la herramienta, ésta puede aprender a detectar el contexto.

Cabe mencionar que las funcionalidades detalladas pueden ser incorporadas como un servicio cloud en la herramienta IBM DOORS Family. Esta última es una solución escalable que tiene por objetivo la gestión de proyectos. En otras palabras, *Engineering Requirements Quality Assistant* es una extensión o complemento de software que puede ser incluida para añadir nuevas funcionalidades y características a otros módulos desarrollados por IBM.

Engineering Requirements Management DOORS es una herramienta de gestión de requerimientos que permite la captura, trazabilidad, análisis y gestión de cambios. Resulta importante mencionar que no incorpora servicios de IA. Entre sus principales características se destacan:

- Disponibilidad de una base de datos integrada para el almacenamiento de requerimientos.
- Herramientas que optimizan la comunicación, trabajo colaborativo y verificación de requerimientos.
- Capacidad de vincular requerimientos a elementos de diseño y casos de prueba para mejorar la trazabilidad.

Múltiples herramientas que contribuyen a la verificación de la calidad han sido revisadas en Konig et al. (2017). Estas herramientas fueron originalmente contempladas para su análisis en el proyecto Proceso de Análisis y modelado de Requerimientos (RAMP) propuesto por Génova et al. (2013) para la eficiencia y calidad de los requerimientos expresados en lenguaje natural durante el desarrollo de sistemas complejos. Principalmente, el estudio focaliza en la revisión de 8 (ocho) herramientas que no incorporan servicios de IA estas son: RQA, LEXIOR, Requirements Assistant, DESIRE, QUARS, TigerPro, ARM y DOORS. A partir de ellas, es posible destacar los siguientes aspectos:

- La mayoría de las herramientas son de uso comercial. Por lo tanto, su adquisición y mantenimiento incurren en un costo significativo para el proyecto de software.
- Limitaciones en cuanto al formato de entrada de los archivos que contienen los requerimientos.
- Inclusión de vocabularios y bolsas de palabras para realizar la verificación de la calidad bajo contexto controlado.
- Implementación de reglas gramaticales definidas manualmente que permiten el análisis sintáctico.
- Las opciones de descarga correspondientes a versiones de prueba de las herramientas no están disponibles. Esto dificulta la elección de una herramienta particular para su incorporación en el proyecto.
- Falta de mantenimiento, es decir, no registran liberación de nuevas versiones.
- Ausencia de información respecto a las características y criterios de calidad abordadas por herramienta.

En síntesis, se observa que algunas de las estrategias utilizadas por las herramientas para abordar la calidad de los requerimientos pueden considerarse obsoletas, o bien, de rendimiento cuestionable respecto a las exigencias que surgen del contexto actual. La razón principal se debe a que estas herramientas no evolucionaron en nuevas versiones con funcionalidades adicionales. De hecho, tampoco han sido enriquecidas a través de la inserción de nuevas tecnologías de mayor alcance que permitan otros enfoques de evaluación.

Por otro lado, también se destaca el reducido abanico de herramientas que incorporan tecnologías de vanguardia para este propósito de estudio. Del mismo modo, es notable la falta de detalle respecto a las técnicas de IA incorporadas en las herramientas. En otras palabras, las características incluidas se abordan superficialmente. Sin dudas, este hecho se relaciona con la competencia global, sin embargo, su integración en los proyectos puede verse afectada dado que su funcionamiento por lo general es percibido como una caja negra.

4.7/ Limitaciones

Tal como se menciona en la sección anterior, las tareas de verificación de requerimientos que aún se realizan manualmente, son consideradas costosas en cuanto a tiempo y propensa a errores. Por este motivo, se propone una solución alternativa que integra tecnologías de IA asociadas a las redes neuronales recurrentes y transformadoras descritas en el Capítulo 3. La herramienta propuesta en este trabajo de Tesis es pionera en la comunidad académica en lo que respecta a la integración de modelos neuronales que resultan de un proceso de entrenamiento previo, como servicios en un entorno web para el soporte a las tareas de verificación de requerimientos de software. Sin embargo, dado que se presenta el despliegue de la primera versión de la herramienta, posee algunos aspectos limitantes en cuanto a su alcance. Estos últimos, se mencionan a continuación:

- Procesamiento de requerimientos en formato CSV.
- Los requerimientos de entrada están circunscritos al idioma inglés. La razón principal de este limitante radica en la disponibilidad de técnicas de procesamiento de lenguaje natural en dicho idioma para su estudio en conjunto a las redes neuronales.
- La configuración de los modelos embebidos se establece de forma predeterminada para su ejecución correspondiente. Por lo tanto, aún no se admiten ajustes sobre los valores de hiperparámetros y métricas de evaluación.

Después de todo lo expuesto, el trabajo futuro estará destinado a la liberación de nuevas versiones de la herramienta con la inclusión de nuevas funcionalidades, tales como el procesamiento de requerimientos en otros idiomas, opciones de generación de conjuntos de entrenamiento propios y configuración avanzada de modelos neuronales que contribuyan a la evaluación automática de la calidad de los requerimientos que con ello, facilitar y agilizar las tareas de verificación de los profesionales del software.

4.8/ Conclusiones

Por lo general, los controles de calidad se realizan en forma manual, lo que constituye una tarea poco trivial que consume tiempo y esfuerzo. Esta tarea, además de ser considerada costosa y propensa a errores, genera largos bucles de retroalimentación (Femmer y Vogelsang, 2018). A saber, cuando un ingeniero de requerimientos solicita una revisión o inspección, esto implica la coordinación de participantes, organización de reuniones, consenso y definición de criterios de evaluación, discusión de los resultados y verificación de los defectos corregidos. En consecuencia, esto ralentiza la liberación de los requerimientos para las fases posteriores de diseño y desarrollo de software. Más aún, si el proyecto de software amplía su tamaño y alcance, dado que también aumenta el volumen de requerimientos por revisar. En relación a esto, el presente capítulo describe el desarrollo de una herramienta que permite la evaluación automática de la calidad de los requerimientos en lenguaje natural. Para ello, los modelos neuronales obtenidos se integran como servicios de inteligencia artificial previamente entrenados en un entorno web. El propósito principal es proveer a los profesionales de software mecanismos automáticos para el control y evaluación de los requerimientos, además de contribuir al desarrollo de software de calidad con tiempos y costos controlados. En otras palabras, se pretende automatizar las tareas de verificación de requerimientos que demandan esfuerzos intensivos y con ello, proveer soluciones que contribuyan a mejorar la calidad del software.

La herramienta es capaz de evaluar de forma automática los requerimientos en lenguaje natural, a través de los modelos de clasificación basados en redes neuronales profundas embebidos en ella. Dichos modelos pueden predecir la calidad de los requerimientos como resultado del proceso de entrenamiento y evaluación realizado previamente. A partir de ello, los profesionales de software que lleven a cabo las tareas de verificación pueden visualizar de forma rápida y sencilla los resultados que retorna la herramienta. Como se menciona en este Capítulo, esta herramienta no tiene por objetivo reemplazar la participación de los profesionales de software en la actividad de validación de requerimientos. De hecho, el flujo de trabajo propuesto incluye el monitoreo de los resultados, a fin de que los ingenieros supervisen los valores que retorna la herramienta y sea posible corregirlos oportunamente. En consecuencia, esta herramienta permite enriquecer el proceso de la IR con flujos de trabajo inteligentes basados en tecnologías más potentes y robustas. Del mismo modo, en este Capítulo se incluyen los principales artefactos correspondientes al modelado del sistema. Por último, se describe el diseño arquitectónico y la infraestructura necesaria para el despliegue de la herramienta propuesta.

En el siguiente Capítulo se presenta un caso de estudio que permite evaluar el rendimiento de la herramienta de soporte a la verificación de los requerimientos de software propuesta en este trabajo de Tesis. También, se provee una descripción detallada de las interacciones necesarias del Ingeniero de Requerimientos con la herramienta QASS para llevar a cabo la evaluación de los requerimientos y con ello, conducir a la obtención de requerimientos de calidad y minimizar las tareas manuales.

4.9/ Summary

Generally, quality controls are carried out manually, which constitutes a non-trivial task that consumes time and effort. This task, in addition to being considered costly and error-prone due to its manual nature, generates lengthy feedback loops (Femmer y Vogelsang, 2018). Specifically, when a requirements engineer requests a review or inspection, it involves coordinating participants, organizing meetings, reaching consensus, defining evaluation criteria, discussing results, and verifying corrected defects. Consequently, this slows down the release of requirements for subsequent phases of software design and development. Moreover, if the software project expands in size and scope, the volume of requirements to be reviewed also increases. In relation to this, this chapter describes the development of a tool that enables the automatic evaluation of the quality of natural language requirements. To achieve this, the obtained neural models are integrated as pre-trained artificial intelligence services within a web environment. The main purpose is to provide software professionals with automated mechanisms for requirement control and evaluation, while contributing to the development of quality software with controlled time and costs. In other words, the aim was to automate the verification tasks that require intensive efforts and provide solutions that contribute to improving software quality.

The tool is capable of automatically evaluating natural language requirements through embedded deep neural network-based classification models. These models can predict the quality of requirements as a result of the previously conducted training and evaluation process. Based on this, software professionals carrying out verification tasks can quickly and easily visualize the results returned by the tool. As mentioned in this chapter, the objective of this tool is not to replace the participation of software professionals in the requirements validation activity. In fact, the proposed workflow includes result monitoring, allowing engineers and analysts to oversee the values returned by the tool and make timely corrections if necessary. Consequently, this tool enriches the Requirements Engineering process with intelligent workflows based on more powerful and robust technologies. Similarly, this chapter includes

the main artifacts corresponding to system modeling. Finally, the architectural design and necessary infrastructure for the deployment of the proposed tool are described.

The next chapter presents a case study that evaluates the performance of the proposed software requirements verification support tool in this thesis work. Additionally, a detailed description of the necessary interactions between the Requirements Engineer and the QASS tool is provided to carry out requirement evaluation and, thereby, achieve quality requirements while minimizing manual tasks.

5

Caso de Estudio

En este capítulo se presenta un caso de estudio asociado a un sistema de servicios bancarios, con el propósito de analizar el comportamiento y evaluar el desempeño de la herramienta de soporte a la verificación de requerimientos propuesta en este trabajo de Tesis. En primer lugar, se describe el conjunto de requerimientos de software sometido a evaluación, a través de los modelos neuronales embebidos en la herramienta. Luego, se detalla el flujo de trabajo que facilita las interacciones entre los profesionales del software y la herramienta para llevar a cabo las actividades de validación de requerimientos de forma automática. Posteriormente, se discuten los resultados obtenidos y se analiza de qué manera la herramienta propuesta contribuye a las tareas de verificación en un proyecto de software.

5.1/ Introducción

Los crecientes avances tecnológicos en lo que refiere a la gestión de información y comunicación ha conducido a inversiones masivas en productos de software que operan en línea. Sin duda, esta situación también ha influido en el sector bancario, especialmente sobre los productos de software de banca doméstica basados en la web o también conocidos por su traducción en inglés, como *Home Banking*, que continúan evolucionando en nuevas versiones a fin de incluir más funcionalidades. La razón principal de esta situación se relaciona con los efectos de la pandemia, los cuales intensificaron la digitalización de las operaciones bancarias. En consecuencia, esto ha generado un fuerte crecimiento en el uso de servicios bancarios remotos y, con ello, la consolidación de la transformación digital de las instituciones financieras se ha convertido en una necesidad (Moșteanu et al., 2020). En relación a esto, las instituciones financieras están rediseñando su estructura operativa e incluyendo nuevas tecnologías para la digitalización y gestión de la información, con el propósito de continuar dando respuestas a las necesidades emergentes que surgen en el contexto actual.

En la actualidad, un gran número de bancos ofrecen diferentes soluciones de Home Banking. El propósito principal es proveer servicios bancarios accesibles a través de Internet por medio de computadoras, tablets o teléfonos celulares. En este sentido, la definición de requerimientos de calidad para la inserción de nuevas funcionalidades y el perfeccionamiento de otras, se convierte en un factor clave de éxito. Esto es así debido a que los requerimientos sirven de guía durante el proyecto de desarrollo de software. Por lo tanto, una correcta definición de los requerimientos que considere el cumplimiento de los criterios de calidad, contribuirá a alcanzar productos de software de calidad, en este caso particular, aquellos que ofrecen servicios de Home Banking. Por todo lo expuesto, es que resulta relevante el estudio de la calidad sobre requerimientos de software involucrados en el diseño y desarrollo de un sistema

de Home Banking.

5.2/ Caso de estudio

Con el propósito de ilustrar el comportamiento de la herramienta Quality Assurance Support System (QASS), se propone someter a evaluación un conjunto de requerimientos extraídos de un documento de requerimientos provisto en Saavedra (2016) (AnexoIV). Este documento especifica los requerimientos necesarios para el desarrollo de algunas funcionalidades que refieren a un sistema de Home Banking. Las principales funcionalidades que incluye este documento se listan a continuación:

- Cuentas Bancarias: gestión de saldos y movimientos.
- Transferencias: a cuentas propias en el mismo banco, a cuentas propias en otros bancos, a cuentas de terceros, administración de cuentas destino de transferencias, resumen de transferencias.
- Pagos: pagar servicios e impuestos, consultas de agenda de pagos, consultas y administración de servicios adheridos, consultas de pagos efectuados.
- Compras: consultas de compras en comercios.
- Pedidos al banco: cheques, boletas de depósito, extractos, seguro contra robo en cajero, tarjetas de crédito adicionales, adelantos de haberes.
- Opciones personales: servicios de mensajes y alertas, datos personales.
- Tarjetas de crédito: consultas de tarjeta de crédito, consultas de límites y disponibles, consultas de cierres y vencimientos, consultas de último resumen, consultas de consumos mensuales.
- Tarjetas de débito: modificación límite de extracción diario en cajeros automáticos.
- Plazos fijos: consultas de plazos y tasas, renovación automática, cambio de cuenta de acreditación.
- Préstamos: solicitudes de préstamos personales pre-aprobados, consultas de préstamos personales.

La Figura 5.1 ilustra el flujo de datos del sistema de Home Banking a fin de proveer una visión general de los requerimientos contemplados en el documento ERS. Este diagrama de flujo tiene por propósito representar los procesos y las principales funciones involucradas en el almacenamiento, manipulación y distribución de datos entre los componentes del sistema y su entorno. Cabe mencionar que los requerimientos sometidos a evaluación en este caso de estudio sólo comprenden las unidades operacionales correspondientes a Gestión de Préstamos y Gestión de Cuentas.

A continuación, la Tabla 5.1 provee la descripción de los requerimientos contemplados en este caso de estudio para su correspondiente evaluación de la calidad.

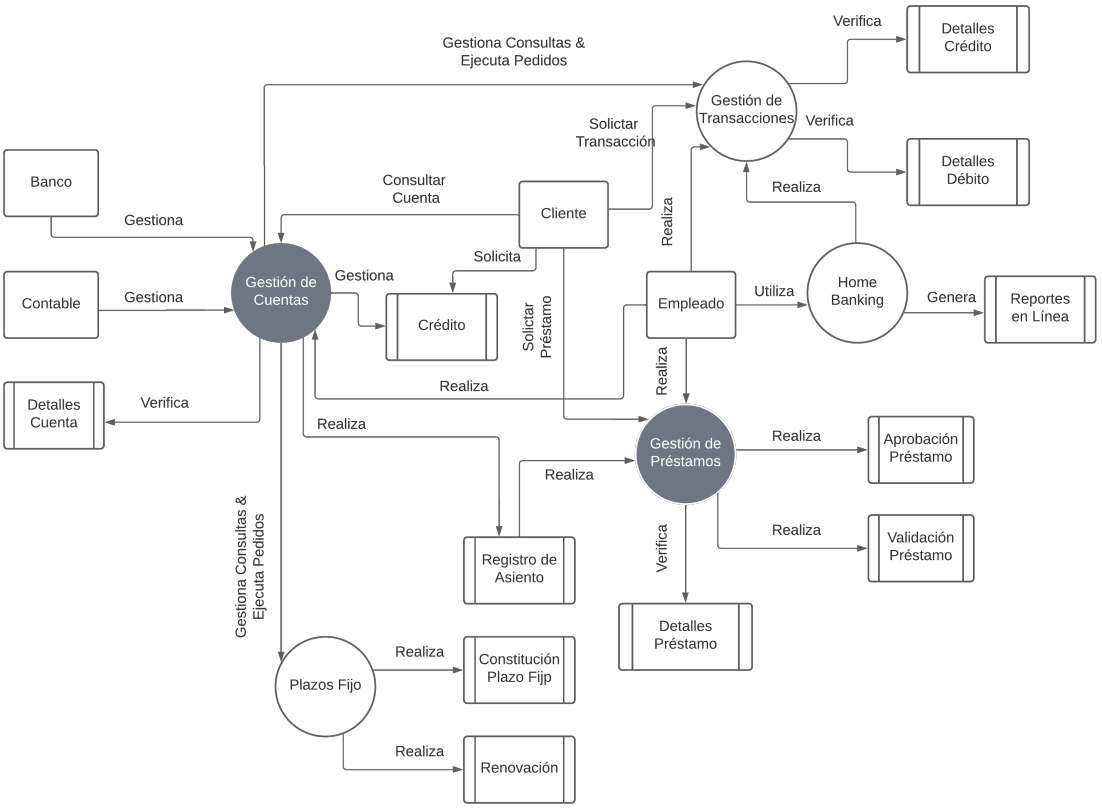


Figura 5.1: Diagrama de flujo de sistema Home Banking.

Tabla 5.1: Requerimientos analizados con QASS.

ID Requerimiento	Descripción
Req. 1	The system shall allow the wire transfer no more than 400 dollars per day.
Req. 2	The system shall allow the user to access to the dashboard using the email or phone number and a password.
Req. 3	The system shall allow the user to print the bank account details.
Req. 4	The system shall allow the user to export the back account details in any format the user want.
Req. 5	The system shall allow the user to export the balance of his/her own account.
Req. 6	The system shall allow the user to export a receipt to prove any update over the account.
Req. 7	The system shall allow the user to query any deposit entries an account have had.
Req. 8	The system shall allow the user to create an additional account.
Req. 9	The system shall allow the user to export a file with details of a bank account statement.
Req. 10	The system shall allow the user to export his/her own account wire transfers details.

5.3/ Flujo de Trabajo

En esta sección se describen las interacciones entre el Ingeniero de Requerimientos y la herramienta QASS para llevar a cabo la evaluación de la calidad de los requerimientos de software seleccionados.

5.3.1/ Ingresar Requerimientos

El Ingeniero de Requerimientos carga el conjunto de requerimientos que será sometido a evaluación, a través de la herramienta QASS. Los requerimientos de software están contenidos en un archivo con formato CSV y están expresados en lenguaje natural en idioma inglés.

5.3.2/ Ajustar Requerimientos

Luego de cargar los requerimientos, éstos pueden visualizarse mediante la interfaz de usuario provista por la herramienta en un formato tabular. Cada requerimiento se identifica con ID (identificador único) y su correspondiente descripción para proveer una mejor organización. La herramienta QASS ofrece las opciones de añadir nuevo, editar y eliminar requerimiento, lo cual provee flexibilidad al ingeniero para realizar ajustes sobre los requerimientos que serán evaluados. Las funciones de ajustes son opcionales, por lo tanto, el ingeniero puede ejecutarlas oportunamente. La Figura 5.2 muestra la interfaz que permite añadir un nuevo requerimiento al conjunto previamente cargado.

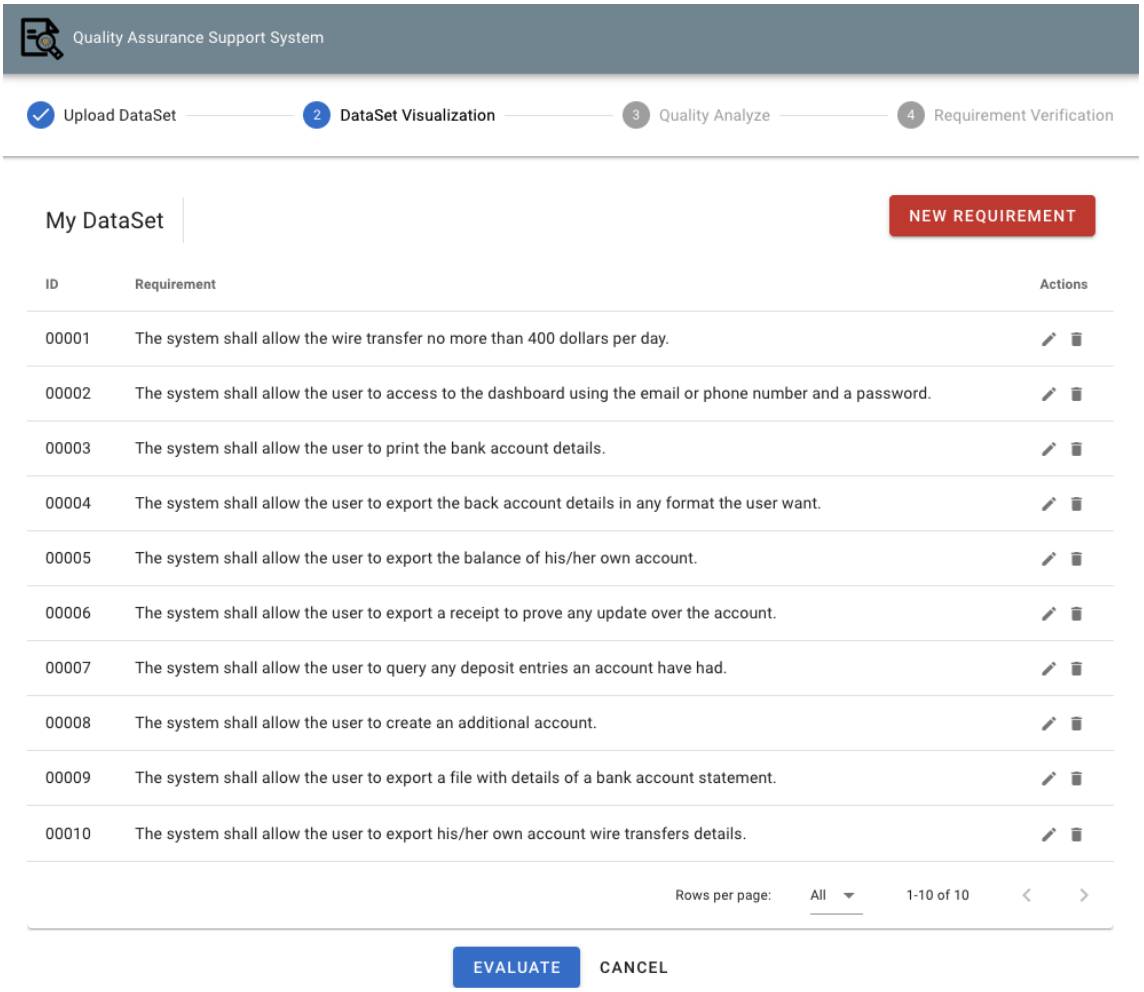


Figura 5.2: Conjunto de Requerimientos.

Como se observa el Ingeniero de Requerimientos puede añadir la descripción de un nuevo requerimiento al presionar el botón *New Requirement*. A continuación, la herramienta despliega una ventana que permite su carga (Figura 5.3). La figura 5.4 presenta el requerimiento añadido, el cual se almacena en la base de datos provista por QASS, junto al conjunto de requerimientos cargados.

5.3.3/ Evaluar Requerimientos

A continuación, los requerimientos se someten a evaluación, a través de los modelos neuronales integrados en la herramienta como servicios. Los requerimientos se evalúan de forma automática cuando el ingeniero presiona el botón *«Evaluate»*. Cabe señalar que los servicios de IA que provee la herramienta con el encapsulamiento de los modelos neuronales, se ejecutan con una configuración predeterminada, es decir, con los valores de parámetros e hiperparámetros establecidos en las fases de entrenamiento y evaluación, detallados en el Capítulo 3. La Figura 5.5 muestra los resultados obtenidos en el proceso de evaluación de requerimientos.

Luego de la ejecución de los servicios de IA para evaluar automáticamente los requerimientos, la herramienta provee los resultados correspondientes en formato tabular. Cada requerimiento sometido

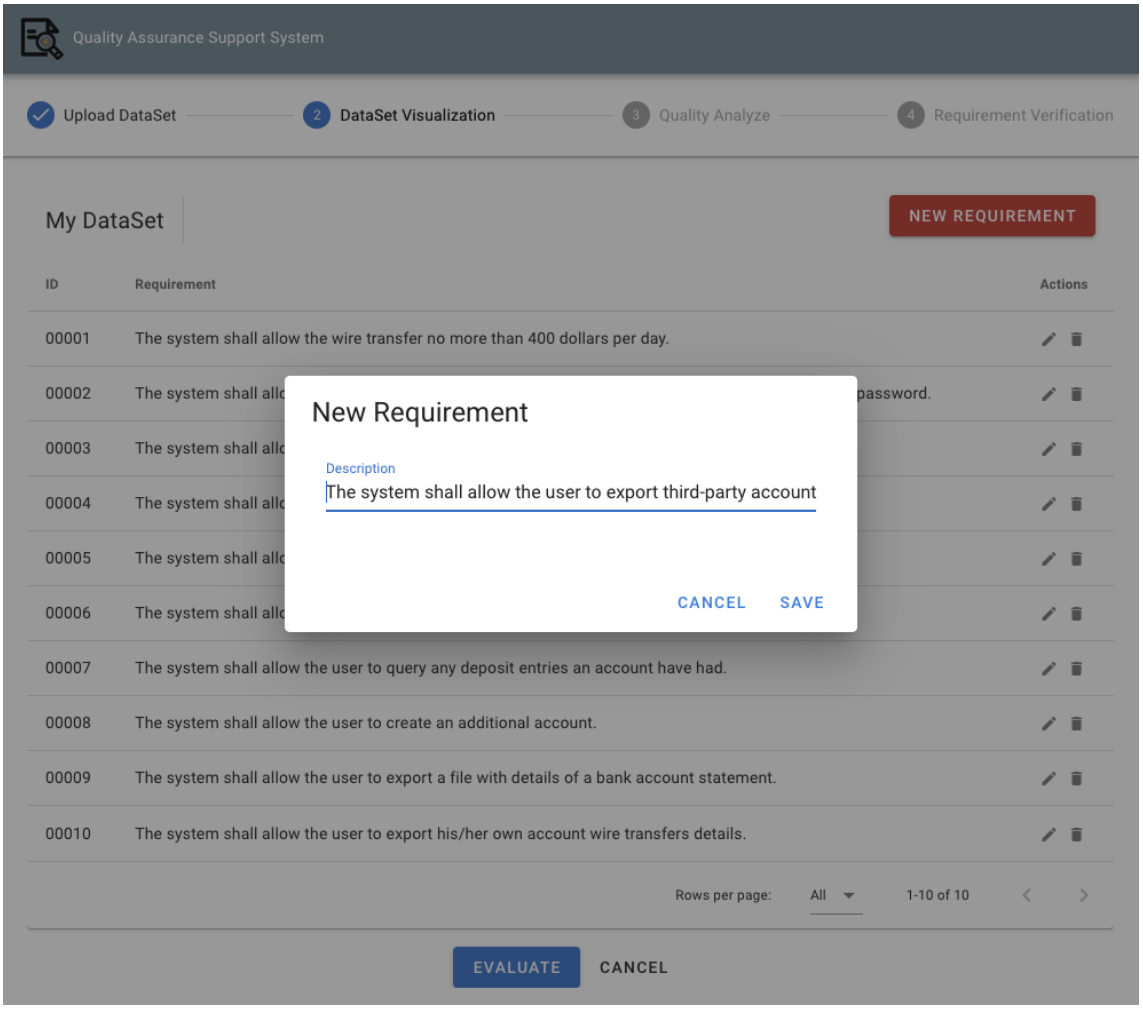


Figura 5.3: Añadir Nuevo Requerimiento.

a evaluación está acompañado de un identificador y su descripción. Por su parte, las características de calidad alcanzadas y/o logradas se indican con variables dicotómicas (Yes/No). La Figura 5.6 muestra los requerimientos que serán verificados por el ingeniero para su posterior edición en caso de que sea necesario.

Particularmente se observa que el requerimiento ***Req 11*** cumple correctamente con las características de calidad evaluadas. Por su parte, ***Req 01*** es clasificado como *Ambiguo*. Posiblemente esto se deba a la presencia de expresiones negativas en su descripción. El ***Req 02*** no cumple con la característica de calidad *Singular*, dado que se percibe la presencia de conectores AND y OR. Esta condición se repite para las características de calidad *Correcto*, *Factible* y *Conforme*. El ***Req 03*** no cumple con las características de calidad *Correcto*, *Completo*, *Verificable* y *Conforme*, dado que la descripción no señala los detalles que se requieren imprimir. En cuanto al ***Req 04***, el modelo detecta que no cumple con las características de calidad *Completo*, *Correcto*, *No Ambiguo*, *Verificable*, *Factible* y *Conforme*. Esto está asociado a la presencia de la palabra «any» en la descripción, dado que no se especifica cuáles son los detalles de la cuenta que se desea exportar y, con ello, genera ambigüedad.

El ***Req 05*** es detectado como un requerimiento *Ambiguo* debido a que su descripción presenta pronombres «his / her». Si bien la presencia de símbolos « / » (barra) puede conducir a la definición

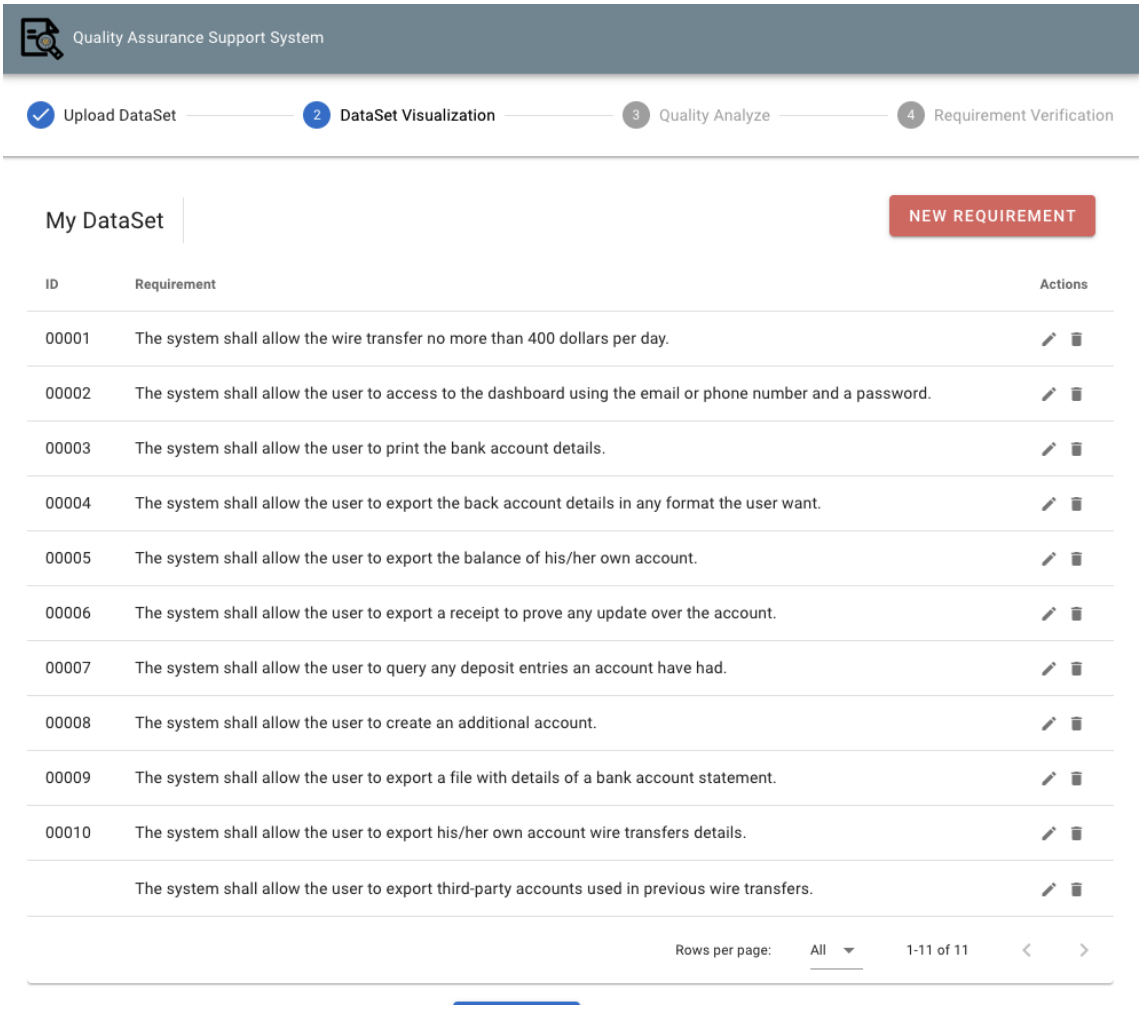


Figura 5.4: Visualizar Requerimientos.

de requerimientos ambiguos, en este caso particular su evaluación podría ser corregida por el ingeniero que lleva a cabo las tareas de verificación.

De acuerdo a la herramienta, el ***Req 06*** no cumple con las características de calidad *Correcto*, *Completo*, *Factible* y *Conforme*. Del mismo modo que el requerimiento anterior, se observa la presencia de la palabra «any» en su definición. En cuanto al ***Req 07***, no cumple con las características de calidad *Verificable* y *Factible*. Sin embargo, la herramienta no detecta el incumplimiento de las características de calidad *Correcto*, *Ambiguo* y *Completo* por lo que su evaluación puede ser corregida por el ingeniero en el paso de verificación. El ***Req 08*** no cumple con las características *Correcto*, *Ambiguo* y *Completo*. Esto es así debido a que es posible que esté asociado a la ausencia de detalle en lo que refiere al tipo de cuenta que el usuario quiere crear.

Por su parte, el ***Req 09*** no cumple con las características de calidad *Correcto*, *Completo*, *Verificable*, *Factible* y *Conforme*, dado que no se especifica la información necesaria sobre los detalles a exportar. Por último, el ***Req 10*** no cumple con las características de calidad *Correcto*, *Completo*, *Verificable*, *Factible* y *Conforme*. También se detecta la presencia de pronombres «his / her» y la información referida a los detalles no es incluida en la descripción.

La herramienta además, permite imprimir el reporte con los resultados obtenidos. El reporte se

genera en formato PDF y con el propósito de mantener la consistencia de los requerimientos, se muestran con el ID y la descripción correspondiente. Por su parte, el cumpliento de las características de calidad se señala mediante variables dicotómicas (Yes/No). Por último, cabe mencionar que la herramienta también es capaz de proveer una representación gráfica de los resultados a través de un gráfico radar y circular, con el propósito de resumir los resultados obtenidos. La Figura 5.5 muestra la interfaz correspondiente a las representaciones gráficas provistas por la herramienta.

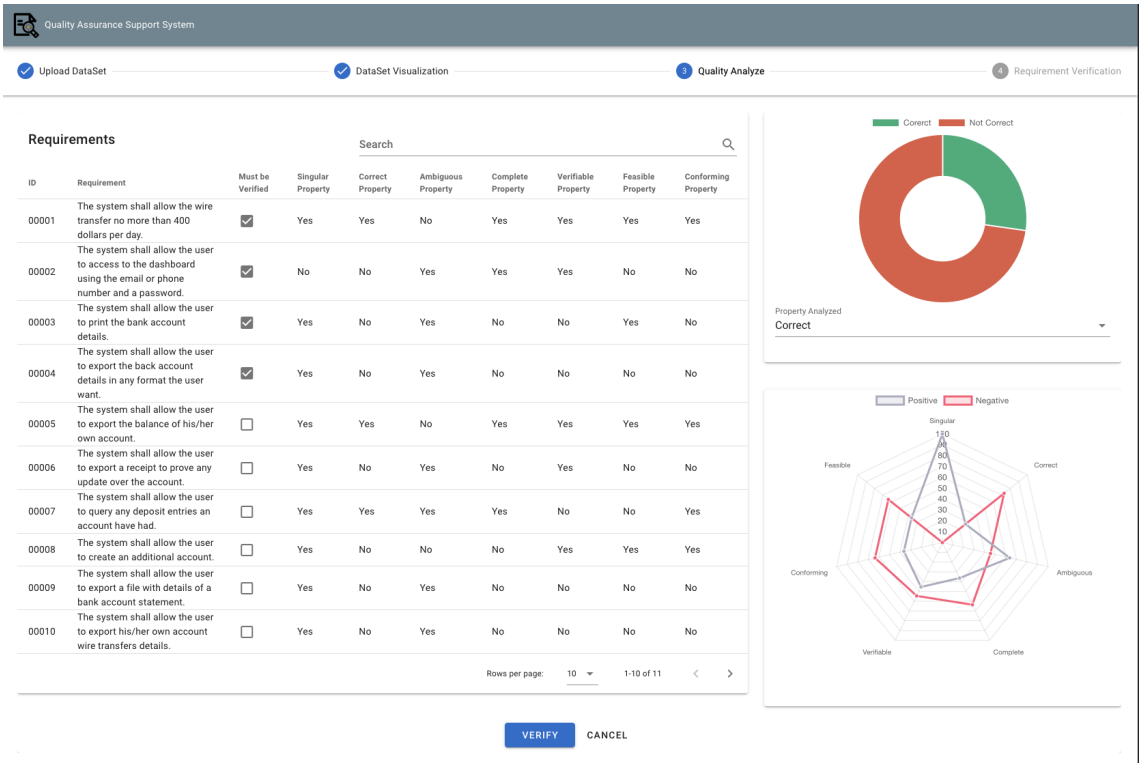


Figura 5.5: Resultados Evaluación de Requerimientos.

5.3.4/ Verificar Requerimientos

El ingeniero es capaz de monitorear los resultados a través de la herramienta y, en base a ello, realizar correcciones oportunas de forma rápida y sencilla. Para lograrlo, el ingeniero debe seleccionar desde la interfaz los requerimientos que desea verificar. En consecuencia, la herramienta provee una interfaz sólo con los requerimientos seleccionados. En esta última, la herramienta permite la corrección de los requerimientos cuyos resultados de evaluación no satisfacen el criterio del ingeniero de requerimientos que lleva a cabo la tarea de verificación. Del mismo modo, QASS permite la eliminación de un requerimiento particular, en caso de ser necesario. La Figura 5.6 ilustra el proceso de verificación de un requerimiento en la herramienta.

Por último, la herramienta QASS también provee la opción de imprimir un reporte en formato PDF con las correcciones realizadas sobre los requerimientos. La Figura 5.7 muestra el reporte generado por la herramienta. Se observa que las predicciones realizadas por la herramienta respecto a las características de calidad se representan con variables dicotómicas (Yes/No). Por otro lado, aquellos requerimientos corregidos se señalan con el ícono de una cruz (X).

Las tareas de verificación y validación buscan comprobar que el software por desarrollar cumpla con

sus especificaciones y brinde las funcionalidades deseadas por el cliente. Dichas tareas de comprobación deberían comenzar tan pronto como estén disponibles los requerimientos y continuar a través de todas las etapas del proceso de desarrollo (Sommerville, 2020).

El propósito de la verificación es comprobar que el software cumpla con los requerimientos establecidos. Por su parte, la validación tiene por objetivo garantizar que el software cumpla con las expectativas del cliente. En otras palabras, busca comprobar la conformidad con la especificación para demostrar que el software hace lo que el cliente espera que haga. Si bien tienen objetivos diferentes, ambas se contemplan en el proceso de la IR a través de las actividades de validación.

Específicamente, a través de la herramienta QASS se ha focalizado en proporcionar soporte a la tareas de verificación de requerimientos. El desarrollo del caso de estudio descrito ha demostrado su utilidad y facilidad de uso. Más aún, se ha evidenciado que la IR puede beneficiarse de nuevas técnicas y métodos a través de la inserción de tecnologías como la IA. Si bien aún quedan desafíos por resolver en el área, esta propuesta puede ser considerada como el inicio de flujo de trabajo inteligente en lo que refiere al desarrollo de productos de software.

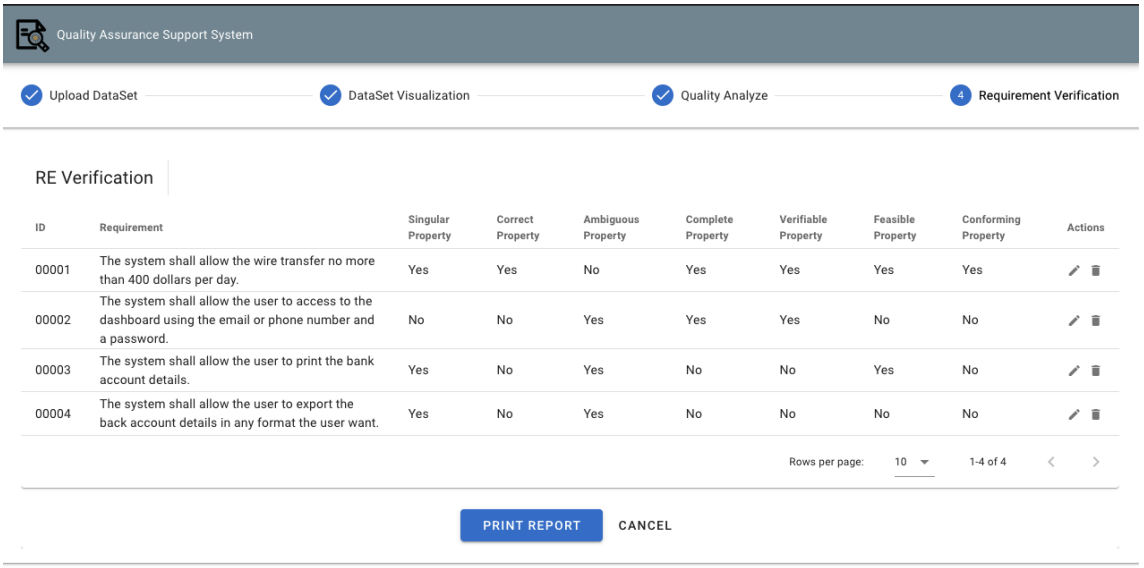
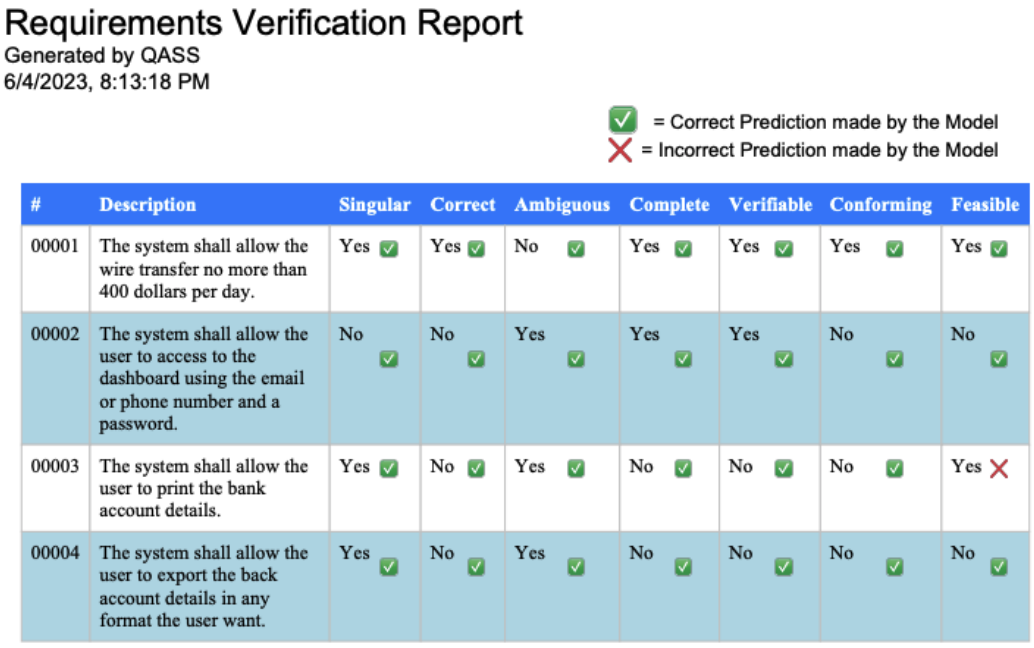


Figura 5.6: Verificación de Requerimientos.

5.4/ Conclusiones

En este capítulo se ha presentado un caso de estudio que describe los requerimientos de software asociados a las unidades operativas Gestión de Préstamos y Gestión de Tarjetas de Débito y Crédito para un sistema de Home Banking. El objetivo principal de este capítulo es ilustrar el comportamiento y desempeño de la herramienta *Quality Assurance Support System* propuesta en este trabajo de Tesis.

La herramienta desarrollada permite al ingeniero cargar los requerimientos que se desea evaluar de forma fácil y rápida. Además, provee los mecanismos necesarios para realizar ajustes sobre los requerimientos previamente cargados mediante las funciones de alta, baja y modificación. Luego, es posible iniciar la evaluación de los requerimientos a través de los servicios de IA que ofrece la herramienta. Esto es factible dado que la herramienta QASS integra los modelos neuronales obtenidos como servicios



Requirements Verification Report
Generated by QASS
6/4/2023, 8:13:18 PM

= Correct Prediction made by the Model
= Incorrect Prediction made by the Model

#	Description	Singular	Correct	Ambiguous	Complete	Verifiable	Conforming	Feasible
00001	The system shall allow the wire transfer no more than 400 dollars per day.	Yes ✓	Yes ✓	No ✓	Yes ✓	Yes ✓	Yes ✓	Yes ✓
00002	The system shall allow the user to access to the dashboard using the email or phone number and a password.	No ✓	No ✓	Yes ✓	Yes ✓	Yes ✓	No ✓	No ✓
00003	The system shall allow the user to print the bank account details.	Yes ✓	No ✓	Yes ✓	No ✓	No ✓	No ✓	Yes ✗
00004	The system shall allow the user to export the back account details in any format the user want.	Yes ✓	No ✓	Yes ✓	No ✓	No ✓	No ✓	No ✓

Figura 5.7: Corrección de resultados.

(Capítulo 3).

Por otro lado, el ingeniero de requerimientos es capaz de llevar a cabo las correcciones pertinentes cuando los resultados que retorna la herramienta no satisfacen los criterios del experto. Cabe mencionar que QASS también contempla el monitoreo y corrección de resultados, dado que tiene por propósito proveer soporte a las tareas de verificación y, no así el reemplazo del ingeniero de requerimientos. La herramienta también proporciona las funcionalidades necesarias para la generación de reportes con los resultados obtenidos.

Por último, el flujo de trabajo establecido para la evaluación de los requerimientos, a través de la herramienta, ha demostrado ser útil y es capaz de proveer soporte al ingeniero durante las tareas de verificación.

En el siguiente Capítulo se presentan las conclusiones generales que surgen a partir del desarrollo de este trabajo de Tesis y, además, se describen futuras líneas de investigación.

5.5/ Summary

In this chapter, a case study has been presented that describes the software requirements associated with the operational units of Loan Management and Debit and Credit Card Management for a Home Banking system. The main objective of this chapter is to illustrate the behavior and performance of the proposed Quality Assurance Support System tool in this thesis work.

The developed tool allows the engineer to easily and quickly load the requirements they wish to evaluate. It also provides the necessary mechanisms for making adjustments to the previously loaded requirements through the functions of addition, deletion, and modification. Subsequently, it is possible to initiate the evaluation of the requirements through the AI services offered by the tool. This is feasible because the QASS tool integrates the obtained neural models as services (Chapter 3).

On the other hand, the requirements engineer is capable of making relevant corrections when the results returned by the tool do not meet the expert's criteria. It should be mentioned that QASS also includes result monitoring and correction, as its purpose is to provide support to the verification tasks rather than replacing the requirements engineer. The tool also provides the necessary functionalities for generating reports with the obtained results.

Lastly, the established workflow for requirement evaluation through the tool has proven to be useful and is capable of providing support to the engineer during verification tasks.

The next chapter presents the general conclusions that arise from the development of this thesis work and also describes future lines of research.

6

Conclusiones Generales y Trabajos Futuros

En este capítulo se provee una síntesis de las principales contribuciones llevadas a cabo en el marco de este trabajo de Tesis. También se describen las conclusiones generales que surgen a partir de la investigación. Por último, se proponen futuras líneas de investigación.

6.1/ Principales Contribuciones

El crecimiento exponencial de las tecnologías de información ha dado lugar al surgimiento de sistemas de software modernos de mayor complejidad y alcance. En virtud de ello, nuevos desafíos han surgido en lo que refiere al desarrollo de software de calidad. La calidad del software es definida como el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las expectativas del cliente. Desde la perspectiva de la Ingeniería de Requerimientos, esto puede ser alcanzado desde etapas tempranas, a través de las características de calidad de los requerimientos, que conducen a la correcta comprensión y especificación de los requerimientos de software. Esto es importante para un proyecto de software, dado que los requerimientos guían de forma transversal el proceso de desarrollo del producto. A fin de enriquecer métodos tradicionales de evaluación de la calidad de los requerimientos, en esta investigación se ha propuesto el desarrollo de un conjunto de clasificadores binarios basados en redes neuronales profundas los cuales, posteriormente, se integran en una herramienta web. Esta última, tiene por objetivo proporcionar soporte a las tareas de verificación. A continuación, se describen brevemente las principales contribuciones llevadas a cabo en cada capítulo que compone la presente Tesis Doctoral.

En el Capítulo 2 se describe la importancia de la obtención de requerimientos de calidad como factor determinante de éxito en un proyecto de desarrollo de software. Particularmente, se abordó el estudio de las características de calidad para requerimientos individuales establecidas en el estándar ISO IEC IEEE 29148 (2018). Entre las múltiples técnicas que permiten la obtención y especificación de requerimientos, en este capítulo se destaca el uso del lenguaje natural. Éste permite llevar a cabo la comunicación y colaboración entre los interesados sin conocimientos técnicos previos. Por esta razón, es que su uso es ampliamente aceptado en la industria del software. Los efectos negativos sobre su aplicación también han sido analizados. La flexibilidad que proporciona el lenguaje natural puede conducir a la especificación de requerimientos ambiguos, vagos y poco claros y, en consecuencia, afectar la calidad de los requerimientos. En este sentido, la comunidad académica ha realizado grandes esfuerzos para abordar diversos desafíos asociados a la calidad de los requerimientos. Particularmente, en esta

investigación se propuso abordar el estudio de la calidad de los requerimientos a través de nuevas técnicas que ofrece la Inteligencia Artificial, con el fin de hacer uso de sus beneficios y enriquecer métodos tradicionales. Para ello, previamente se llevó a cabo una revisión sistemática de la literatura que ha permitido evidenciar su uso en el área de requerimientos.

En el Capítulo 3 se propone la evaluación de las características de calidad de los requerimientos de software a través de la aplicación de redes neuronales profundas y técnicas de procesamiento de lenguaje natural. Para lograrlo, en primer lugar, se llevó a cabo la construcción de un corpus de requerimientos. Éste se compone de requerimientos individuales escritos en inglés que cubren múltiples dominios procedentes de diversas compañías y proyectos universitarios. Luego, se exploró sobre el uso de redes neuronales recurrentes y transformadoras. Por lo tanto, se incluye la descripción de la configuración experimental para ambos tipos de redes neuronales. Múltiples variantes de redes neuronales recurrentes fueron evaluadas LSTM, BiLSTM, GRU, BiGRU. En lo que refiere a las redes neuronales transformadoras, se abordó la aplicación de BERT. La ejecución de los modelos neuronales obtenidos ha permitido analizar el desempeño de ambas arquitecturas y, con ello, determinar cuál resulta adecuada para el estudio de la calidad. Los resultados obtenidos demuestran que las redes neuronales recurrentes presentan un rendimiento sustancialmente superior a las transformadoras para la evaluación de la calidad de los requerimientos en lenguaje natural.

En el Capítulo 4 se describe el desarrollo de una herramienta web que ha sido denominada *Quality Assurance Support System*. La herramienta integra los modelos neuronales obtenidos como servicios, lo cual permite a la Ingeniería de Requerimientos beneficiarse de las tecnologías de Inteligencia Artificial. En efecto, a través de la herramienta es posible llevar a cabo la evaluación automática de la calidad de los requerimientos utilizando los modelos propuestos en el Capítulo 3. Las funcionalidades incluidas refieren a la carga, ajustes, evaluación y verificación de requerimientos.

La herramienta desarrollada también contempla la posibilidad de realizar correcciones de forma oportuna si los resultados provistos no satisfacen los criterios de los ingenieros. Del mismo modo, el ingeniero es capaz de generar reportes con los resultados obtenidos en la evaluación de los requerimientos. Una característica clave de la herramienta, es que su uso no requiere de conocimientos previos relacionados al aprendizaje automático. En cuanto a sus características de diseño, se destacan los siguientes aspectos: uso intuitivo, fácil y rápido. Por último, la herramienta ha sido diseñada como una solución escalable, por lo tanto, nuevas funcionalidades podrán añadirse en versiones futuras.

En el Capítulo 5 se presenta un caso de estudio que permite describir las interacciones entre el ingeniero y la herramienta propuesta. Específicamente, se abordó la evaluación de un conjunto de requerimientos involucrados en el desarrollo de un sistema de Home Banking. Los resultados obtenidos han demostrado que su integración en el proceso de la Ingeniería de Requerimientos es factible y que, de hecho, su uso no demanda esfuerzos y trabajo adicional.

6.2/ Conclusiones Generales

Este trabajo de Tesis permite evidenciar la factibilidad asociada a la aplicación de redes neuronales profundas para el estudio de la calidad de los requerimientos. Es importante destacar que no sólo se llevó a cabo la construcción de un corpus de requerimientos y múltiples configuraciones experimentales para la definición de modelos de clasificación basados en redes neuronales que permiten evaluar las características de calidad de requerimientos individuales. También se presentó el desarrollo de una herramienta

innovadora que integra estos modelos neuronales para dar soporte a las tareas de verificación en el proceso de la Ingeniería de Requerimientos. En relación a esto, no se hallaron propuestas académicas que proporcionen herramientas con características similares para abordar el análisis de la calidad.

En cuanto a los resultados obtenidos, se observa que las redes neuronales recurrentes resultan adecuadas para evaluar las características de calidad. En consecuencia, esto motiva a seguir investigando sobre la aplicación de otras posibles variantes, a fin de explorar y evaluar nuevas arquitecturas.

6.3/ Trabajos Futuros

A partir de este trabajo de Tesis se plantean diferentes líneas de investigación a futuro, las cuales pueden agruparse en cuatro ejes temáticos que se detallan a continuación.

6.3.1/ Influencias entre Características de Calidad

Luego de analizar en detalle las características de calidad sugeridas por el estándar ISO IEC IEEE 29148 (2018) para requerimientos individuales de software, resulta importante destacar las relaciones que surgen entre las mismas. El estudio de las relaciones entre características de calidad ha sido introducido por Génova et al. (2013) y posteriormente, abordadas por otros investigadores de la comunidad académica. Inicialmente, éstos proponen una taxonomía de características de calidad bajo dos perspectivas diferentes: (a) *Perspectiva del Cliente*, considera aquellas características de calidad que permiten comprobar la correctitud de las necesidades solicitadas por el cliente y (b) *Perspectiva del Ingeniero de Requerimientos*, incluye las características de calidad que permiten determinar que los requerimientos especificados sean consistentes con las necesidades del cliente, lo cual posteriormente conducirá al desarrollo y mantenimiento del producto de software. El análisis propuesto comprende las características de calidad, *Consistente*, *Completo*, *Entendible*, *Trazable*, *Preciso*, *Atómico*, *Abstracto*, *Verificable*, *Validable*, *Modificable* y *No ambiguo*, incluidas en un modelo de calidad.

Por su parte, Saavedra et al. (2013) señalan que la obtención de una característica de calidad en particular puede impactar sobre la consecución de otras. En efecto, esto ha dado lugar al surgimiento de dependencias positivas y negativas entre características de calidad. Esta investigación aborda el estudio de las características de calidad, *Modificable*, *Verificable*, *Correcto*, *Conciso*, *Referencias cruzadas*, *Organizado*, *Almacenado electrónicamente*, *No redundante*, *Atómico*, *Trazable*, *No ambiguo*, *Preciso*, *Entendible*, *Consistente*, *Completo*, *Reutilizable*, *Actualizado*, *Correcto nivel de detalle*, *Anotado*, *Interpretable* y *Actualizado*, también incluidas en un modelo de calidad propio. Sin embargo, el análisis provisto por ambas investigaciones es cuestionado dado que sólo se realiza desde una perspectiva lógica y meramente intuitiva que carece de evidencia empírica (Kummler, 2021).

Dado que este trabajo de Tesis está sujeto a las características establecidas por el estándar ISO IEC IEEE 29148 (2018), resulta apropiado ajustar el análisis de influencias para aquellas características que constituyen el modelo de calidad incluido en dicho estándar. El modelo de calidad provisto procura normalizar las características de calidad que deben considerarse para garantizar la calidad de los requerimientos. Esto último, es indispensable para la industria del software. Más aún, si se consideran los múltiples modelos de calidad presentes en la literatura. Dada la diversidad de modelos propuestos, las características de calidad, por lo general, están sujetas a variaciones en lo que refiere a denominación y alcance. Por lo tanto, resulta conveniente el uso de un modelo de calidad estandarizado para llevar

a cabo el análisis de influencias. Los resultados reportados en las investigaciones anteriores sobre las influencias entre características fueron sintetizados y representados a través de un grafo dirigido. Cabe destacar que este grafo aborda sólo las características de calidad contempladas en el estándar antes mencionado (Figura 6.1).

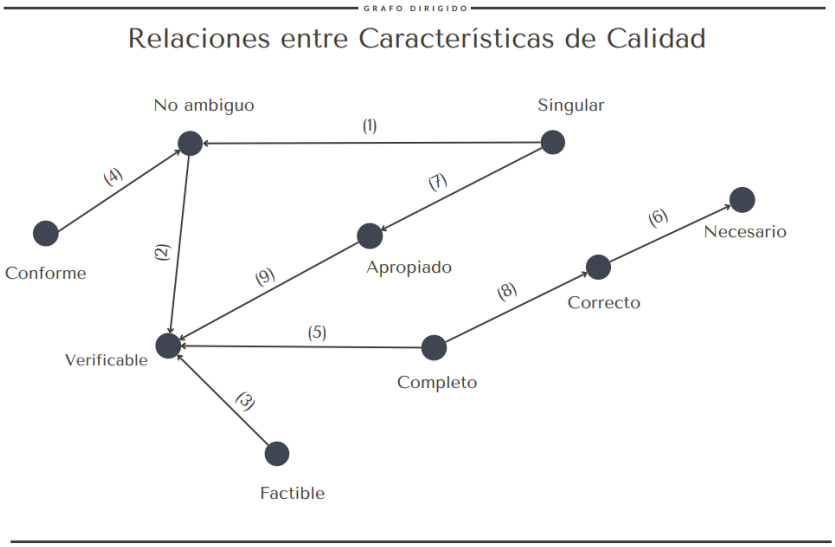


Figura 6.1: Relaciones entre características de calidad.

En primer lugar, se identifican las relaciones positivas entre las características de calidad, es decir, aquellas que surgen cuando el cumplimiento de una característica particular promueve la obtención de otra característica. Luego, se distinguen las relaciones negativas entre características de calidad. Esto refiere a aquellas que surgen cuando cierta característica de calidad no se cumple y, con ello, afecta la obtención de otra característica. Ambos tipos de relaciones se sintetizan a continuación:

Relaciones Positivas

- La ausencia de detalles técnicos excesivos en la descripción de un requerimiento de software reduce la ambigüedad y, en consecuencia, contribuye con la verificabilidad [1-2].
- Un requerimiento conforme que adopta un formato y estilo de escritura adecuado puede disminuir la ambigüedad y, en efecto, facilitar la verificabilidad [4-2].
- La independencia del diseño facilita la verificación de los requerimientos [9].

Relaciones Negativas

- Aquellos requerimientos vagos que presentan términos ambiguos resultan difíciles de verificar [2].
- Los requerimientos no factibles difícilmente serán verificables, puesto que no se puede verificar aquello que no se puede desarrollar [3].
- Los requerimientos incompletos conducen a requerimientos no verificables [5].
- Un requerimiento incorrecto que no representa la necesidad que precisa el cliente, resulta innecesario [6].

- Un requerimiento excesivamente detallado conlleva a incluir posibles soluciones técnicas que afectan a la característica de calidad apropiado [7].
- Un requerimiento incompleto en cuanto a objetivos y restricciones respecto al dominio del problema conduce a requerimientos incorrectos [8].

Luego de identificar las relaciones entre las características de calidad, es crucial evidenciar las mismas de forma empírica. En contraste a los enfoques anteriores basados en una lógica y perspectiva meramente intuitiva, en esta investigación se propone implementar el análisis de correspondencias múltiples (MCA) para tal fin. El MCA, es una herramienta de análisis relacional que permite proyectar individuos y variables en un plano factorial, lo que facilita la identificación y visualización de dependencias mutuas entre categorías. Su ventaja radica en su capacidad de compactar datos en dimensiones reducidas, permitiendo una representación gráfica y análisis detallado de las relaciones. El objetivo principal al emplear el MCA es proporcionar evidencia estadística del estudio de las correlaciones de las características de calidad. Esto no sólo ratificará la presencia de las relaciones identificadas sino que también permitirá una mayor comprensión de las mismas. Esta perspectiva estadística fortalece el análisis, ofreciendo una visión fundamentada y rigurosa de las influencias entre características de calidad, algo que ha sido una brecha en investigaciones anteriores.

6.3.2/ Modelos de Clasificación de Múltiples Etiquetas

En esta sección se propone continuar con el estudio de la calidad de los requerimientos expresados en lenguaje natural, pero en esta oportunidad desde una nueva perspectiva. Esto quiere decir, contemplando las relaciones entre las características de calidad, una vez constatadas por el análisis de correspondencias múltiples propuesto en la sección anterior. Para ello, se propone la construcción de un modelo de clasificación de etiquetas múltiples basado en aprendizaje profundo. Este último deberá ser capaz de comprender y/o aprender que un requerimiento de software (instancia), puede lograr el cumplimiento de una a muchas características de calidad (etiquetas) simultáneamente.

La clasificación de múltiples etiquetas es una variante de los problemas de clasificación, donde es posible asignar múltiples etiquetas a cada instancia. En este tipo de variante no hay restricciones sobre cuántas clases se pueden asignar a una instancia. En contraste a la clasificación multiclase donde, si bien hay varias clases (etiquetas) a las que puede pertenecer una instancia, esta última sólo puede asignarse a una de las clases debido a su condición de exclusión mutua (Lv et al., 2021; Bogatinovski et al., 2022).

6.3.3/ Evaluación de Nuevas Variantes de Redes Neuronales Profundas

En relación a esta línea de investigación, se propone explorar el rendimiento de nuevas variantes de redes neuronales profundas orientadas a tareas de procesamiento de lenguaje natural. El propósito principal es determinar el tipo de red neuronal que resulta más conveniente para evaluar la calidad de los requerimientos. Para ello, resulta necesario estudiar experimentalmente con diversas arquitecturas de redes neuronales.

De forma preliminar, se pretende abordar el estudio de las redes neuronales convolucionales. Estas redes neuronales fueron originalmente diseñadas para el análisis de imágenes (LeCun et al., 1998).

Sin embargo, recientemente se ha puesto en evidencia su capacidad para el análisis de datos secuenciales, entre ellos, el procesamiento del lenguaje natural (Young et al., 2018; Kalchbrenner et al., 2014). Investigaciones recientes demuestran sus aplicaciones en el área de Ingeniería de Requerimientos. Específicamente, se observan estudios en tareas relacionadas con la clasificación de requerimientos y generación de pruebas (Navarro-Almanza et al., 2017; Pudlitz et al., 2019; Baker et al., 2019; Rahman et al., 2019). Los resultados provistos resultan prometedores, razón que motiva el estudio de las redes neuronales convolucionales en lo que refiere al aseguramiento de la calidad de los requerimientos en lenguaje natural.

6.3.4/ Despliegue de Nuevas Versiones de la Herramienta

Para esta línea de investigación se contempla la evolución de la herramienta QASS, a través del desarrollo de nuevos módulos que permitirán el despliegue de versiones más robustas y completas en cuanto a utilidades. El objetivo es incluir nuevas funcionalidades, a fin de enriquecer y fortalecer la herramienta web propuesta. Las principales funcionalidades que se pretenden abordar son las que refieren a:

- *Generación de conjuntos de datos*, consiste en proveer a aquellos usuarios avanzados los mecanismos necesarios para generar conjuntos de datos propios que podrán ser almacenados en las bases de datos provistas por la herramienta. Luego, tales conjuntos de datos podrán entrenarse, a través de los servicios integrados de IA. Esta funcionalidad también comprende la partición del conjunto de datos, es decir, el usuario podrá particionarlo convenientemente para las fases de entrenamiento y prueba.
- *Configuración de Modelo Neuronal con Opciones avanzadas*, consiste en proveer opciones de configuración a los usuarios con conocimientos en aprendizaje automático para realizar ajustes sobre los valores de los hiperparámetros con los que el modelo neuronal será ejecutado. El propósito es proveer un rango de valores posibles sobre *Dropout*, *Hidden Units*, *Learning Rate* y *Epochs* que el usuario podrá establecer para la posterior ejecución del modelo neuronal. Esta funcionalidad también incluirá las opciones habilitar/deshabilitar asociadas a las métricas integradas en la herramienta. En otras palabras, el usuario podrá seleccionar entre las métricas *Accuracy*, *Recall*, *F1-score*, *Precision* con la que evaluará el resultado provisto por el modelo neuronal.

Por último, la Figura 6.2 presenta el diagrama de casos de uso. Éste permite visualizar las interacciones entre los usuarios y el sistema propuesto que incluye las nuevas funcionalidades detalladas anteriormente.

6.4/ Summary

This chapter summarizes the main contributions achieved in this thesis. It also describes the general conclusions reached as a result of the research. Finally, future lines of research are described.

This thesis work provides evidence of the feasibility associated with the application of deep neural networks for studying the quality of requirements. It is important to mention that not only has the development of a corpus of requirements and multiple experimental setups for defining classification models to evaluate the quality characteristics of individual requirements been performed but also the presentation of an innovative tool that integrates these neural models to support verification tasks in the Requirements Engineering process. No academic proposals with similar characteristics to address

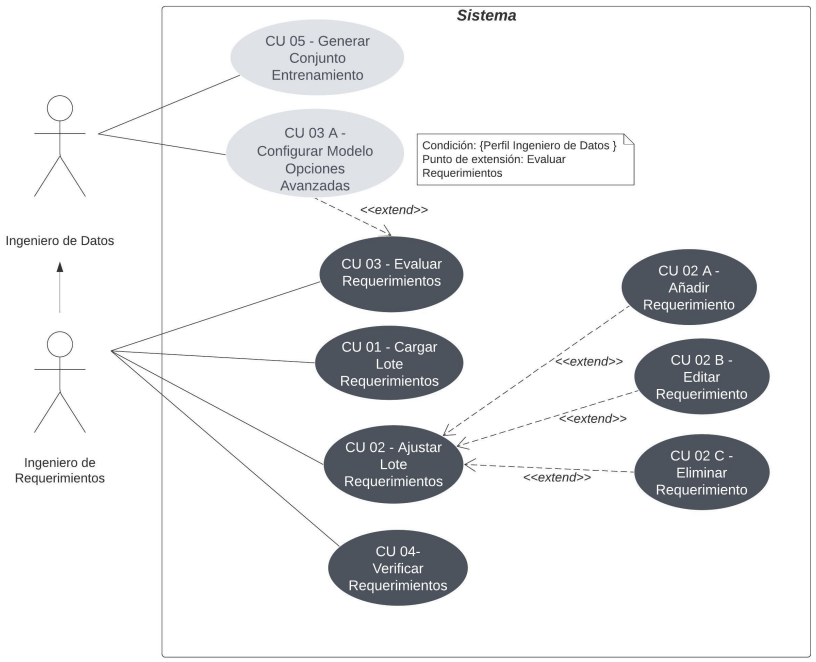


Figura 6.2: Diagrama de casos de uso extendido.

quality were found. Regarding the obtained results, it is observed that recurrent neural networks are suitable for evaluating quality characteristics. Consequently, this motivates further research on the application of other possible variants to explore and assess new architectures.

6.4.1/ Future Works

The future lines of research arising from this thesis propose considering four topics, which are detailed below.

As a future line of research, it is proposed to demonstrate the mutual dependencies manifested between the quality characteristics of the requirements. For this purpose, the use of multiple correspondence analysis is suggested as a relational analysis tool. It projects individuals and variables, in this case, software requirements and quality characteristics, onto a factorial plane.

Consequently, this analysis will allow for the examination and visualization of the relationships that arise between quality characteristics from a statistical perspective, considering the corpus of requirements provided in this research.

In order to introduce this line of research, an analysis of the relationships between the quality characteristics defined by the current standard have been performed (Figure 6.1).

Positive Relationships

- The absence of excessive technical details in the description of a software requirement reduces ambiguity and, consequently, contributes to verifiability [1-2].
- Conforming requirements that adopt an appropriate format and writing style can reduce ambiguity and facilitate verifiability [4-2].

- Design independence facilitates requirements verification [9].

Negative Relationships

- Vague requirements with ambiguous terminology are difficult to verify [2]. Unfeasible requirements will be difficult to verify since it is not possible to verify what cannot be developed [3].
- Incomplete requirements lead to unverifiable requirements [5].
- Incorrect requirements that do not represent the customer's needs are unnecessary [6].
- An excessively detailed requirement leads to the inclusion of possible technical solutions that affect the appropriate quality characteristic [7].
- An incomplete requirement, in terms of objectives and constraints regarding the problem domain, leads to incorrect requirements [8].

Once the relationships between the quality characteristics have been identified, it is possible to conduct a multiple correspondence analysis.

The objective is to provide statistical evidence by studying the correlations among the quality characteristics and, thus, confirm the presence of these relationships and contribute to their understanding and visualization.

6.4.2/ Multiple Label Classification Models

In this section, we propose to continue studying the quality of requirements expressed in natural language but from a new perspective. This involves considering the relationships between the quality characteristics, as verified by the multiple correspondence analysis proposed in the previous section.

Therefore, we propose the development of a multi-label classification model. This model should be capable of understanding and/or learning that a software requirement (instance) can simultaneously comply with one or many quality characteristics (labels).

Multi-label classification is a variant of classification problems where multiple labels can be assigned to each instance. In this type of variant, there are no restrictions on the number of classes that can be assigned to an instance. This is in contrast to multiclass classification, where an instance can only be assigned to one of the classes due to their mutually exclusive condition (Lv et al., 2021; Bogatinovski et al., 2022).

6.4.3/ Application of New Neural Network Variants

Regarding this line of research, it is proposed to explore the performance of new variants of deep neural networks focused on natural language processing tasks. The main objective is to determine the most suitable type of neural network for assessing the quality of requirements. To achieve this, it is necessary to experiment with different neural network architectures.

As a preliminary step, the study will focus on convolutional neural networks. Although originally designed for image analysis, their capability for sequential data analysis, including natural language processing, has recently been highlighted (Young et al., 2018). Recent research demonstrates their

applications in the field of Requirements Engineering, specifically in tasks such as requirements classification and test generation (Navarro-Almanza et al., 2017; Pudlitz et al., 2019; Baker et al., 2019; Rahman et al., 2019). The results contributed by these studies are promising, which motivates further exploration of convolutional neural networks in the context of quality assurance for requirements.

6.4.4/ Deployment of New Versions

This line of research involves the evolution of the QASS tool through the development of new modules that will enable the deployment of more robust and comprehensive versions.

The goal is to incorporate new functionalities to enrich and enhance the proposed web tool. The main functionalities to be addressed are as follows:

- *Dataset Generation:* This functionality aims to provide advanced users with the necessary mechanisms to generate their own datasets, which can be stored in the tool's provided databases. These datasets can subsequently be trained using integrated Artificial Intelligence services. Additionally, this functionality includes dataset partitioning, allowing users to conveniently divide the dataset for training and testing phases.
- *Advanced Neural Model Configuration:* This functionality aims to provide users with machine learning knowledge and the ability to adjust hyperparameter values for the neural model. Users will have options to configure values such as Dropout, Hidden Units, Learning Rate, and Epochs for the subsequent execution of the neural model. Furthermore, this functionality will also allow users to enable or disable the integrated metrics within the tool. Users will be able to select metrics such as Accuracy, Recall, F1-score, and Precision to evaluate the results provided by the neural model.

Bibliografía

[Abad et al. 2017] Abad, Zahra Shakeri H. ; Karras, Oliver ; Ghazi, Parisa ; Glinz, Martin ; Ruhe, Guenther ; Schneider, Kurt: **"What works better? a study of classifying requirements"**. En *2017 IEEE 25th International Requirements Engineering Conference (RE)* IEEE (evento), 2017, páginas 496–501

[Abdukalykov et al. 2011] Abdukalykov, Rolan ; Hussain, Ishrar ; Kassab, Mohamad ; Ormandjieva, Olga: **"Quantifying the impact of different non-functional requirements and problem domains on software effort estimation"**. En *2011 Ninth International Conference on Software Engineering Research, Management and Applications* IEEE (evento), 2011, páginas 158–165

[del Águila y Del Sagrado 2011] Águila, Isabel M. del ; Del Sagrado, Jose: **"Requirement risk level forecast using Bayesian networks classifiers"**. En *International Journal of Software Engineering and Knowledge Engineering* 21 (2011), número 02, páginas 167–190

[Al Sharou et al. 2021] Al Sharou, Khetam ; Li, Zhenhao ; Specia, Lucia: **"Towards a better understanding of noise in natural language processing"**. En *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, 2021, páginas 53–62

[Amershi et al. 2019] Amershi, Saleema ; Begel, Andrew ; Bird, Christian ; DeLine, Robert ; Gall, Harald ; Kamar, Ece ; Nagappan, Nachiappan ; Nushi, Besmira ; Zimmermann, Thomas: **"Software engineering for machine learning: A case study"**. En *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* IEEE (evento), 2019, páginas 291–300

[Arellano et al. 2015] Arellano, Andres ; Zontek-Carney, Edward ; Austin, Mark A.: **"Frameworks for natural language processing of textual requirements"**. En *International Journal On Advances in Systems and Measurements* 8 (2015), páginas 230–240

[Ashrafi 2003] Ashrafi, Noushin: **"The impact of software process improvement on quality: in theory and practice"**. En *Information & Management* 40 (2003), número 7, páginas 677–690

[Atas et al. 2018] Atas, Muesluem ; Samer, Ralph ; Felfernig, Alexander: **"Automated identification of type-specific dependencies between requirements"**. En *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* IEEE (evento), 2018, páginas 688–695

[Baker et al. 2019] Baker, Cody ; Deng, Lin ; Chakraborty, Suranjan ; Dehlinger, Josh: **"Automatic multi-class non-functional software requirements classification using neural networks"**. En *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)* Volumen 2 IEEE (evento), 2019, páginas 610–615

[Bhatia et al. 2013] Bhatia, Jaspreet ; Sharma, Richa ; Biswas, Kanad K. ; Ghaisas, Smita: **"Using grammatical knowledge patterns for structuring requirements specifications"**. En *2013 3rd International Workshop on Requirements Patterns (RePa)* IEEE (evento), 2013, páginas 31–34

[Bogatinovski et al. 2022] Bogatinovski, Jasmin ; Todorovski, Ljupčo ; Džeroski, Sašo ; Kocev, Dragi: **"Explaining the performance of multilabel classification methods with data set properties"**. En *International Journal of Intelligent Systems* (2022)

[Braşoveanu y Andonie 2020] Braşoveanu, Adrian M. ; Andonie, Răzvan: **"Visualizing transformers for nlp: a brief survey"**. En *2020 24th International Conference Information Visualisation (IV)* IEEE (evento), 2020, páginas 270–279

[Cleland-Huang et al. 2010] Cleland-Huang, Jane ; Czauderna, Adam ; Gibiec, Marek ; Emenecker, John: **"A machine learning approach for tracing regulatory codes to product specific requirements"**. En *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, 2010, páginas 155–164

[Dalpiaz et al. 2018] Dalpiaz, Fabiano ; Ferrari, Alessio ; Franch, Xavier ; Palomares, Cristina: **"Natural language processing for requirements engineering: The best is yet to come"**. En *IEEE software* 35 (2018), número 5, páginas 115–119

[Dargan et al. 2016] Dargan, John L. ; Wasek, James S. ; Campos-Nanez, Enrique: **"Systems performance prediction using requirements quality attributes classification"**. En *Requirements Engineering* 21 (2016), número 4, páginas 553–572

[Davis et al. 1993] Davis, Alan ; Overmyer, Scott ; Jordan, Kathleen ; Caruso, Joseph ; Dandashi, Fatma ; Dinh, Anhtuan ; Kincaid, Gary ; Ledeboer, Glen ; Reynolds, Patricia ; Sitaram, Pradip ; others: **"Identifying and measuring quality in a software requirements specification"**. En *[1993] Proceedings First International Software Metrics Symposium* Ieee (evento), 1993, páginas 141–152

[Dekhtyar y Fong 2017] Dekhtyar, Alex ; Fong, Vivian: **"Re data challenge: Requirements identification with word2vec and tensorflow"**. En *2017 IEEE 25th International Requirements Engineering Conference (RE)* IEEE (evento), 2017, páginas 484–489

[Denger et al. 2003] Denger, Christian ; Berry, Daniel M. ; Kamsties, Erik: **"Higher quality requirements specifications through natural language patterns"**. En *Proceedings 2003 Symposium on Security and Privacy* IEEE (evento), 2003, páginas 80–90

[Denger y Olsson 2005] Denger, Christian ; Olsson, Thomas: **"Quality assurance in requirements engineering"**. En *Engineering and managing software requirements*. Springer, 2005, páginas 163–185

[Fabbrini et al. 2001a] Fabbrini, Fabrizio ; Fusani, Mario ; Gnesi, Stefania ; Lami, Giuseppe: **"An automatic quality evaluation for natural language requirements"**. En *Proceedings of the Seventh International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ* Volumen 1 Citeseer (evento), 2001, páginas 4–5

[Fabbrini et al. 2001b] Fabbrini, Fabrizio ; Fusani, Mario ; Gnesi, Stefania ; Lami, Giuseppe: **"The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool"**. En *proceedings 26th annual NASA Goddard software engineering workshop* IEEE (evento), 2001, páginas 97–105

[Feldt et al. 2018] Feldt, Robert ; Oliveira Neto, Francisco G. de ; Torkar, Richard: **"Ways of applying artificial intelligence in software engineering"**. En *2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)* IEEE (evento), 2018, páginas 35–41

[Femmer 2013] Femmer, Henning: **"Reviewing Natural Language Requirements with Requirements Smells–A Research Proposal–"**. En *Proceedings of IDoESE* (2013)

[Femmer y Vogelsang 2018] Femmer, Henning ; Vogelsang, Andreas: **"Requirements quality is quality in use"**. En *IEEE Software* 36 (2018), número 3, páginas 83–91

[Ferrari et al. 2017] Ferrari, Alessio ; Spagnolo, Giorgio O. ; Gnesi, Stefania: **"Pure: A dataset of public requirements documents"**. En *2017 IEEE 25th International Requirements Engineering Conference (RE)* IEEE (evento), 2017, páginas 502–505

[Fitzgerald et al. 2012] Fitzgerald, Camilo ; Letier, Emmanuel ; Finkelstein, Anthony: **"Early failure prediction in feature request management systems: an extended study"**. En *Requirements Engineering* 17 (2012), número 2, páginas 117–132

[Génova et al. 2013] Génova, Gonzalo ; Fuentes, José M ; Llorens, Juan ; Hurtado, Omar ; Moreno, Valentín: **"A framework to measure and improve the quality of textual requirements"**. En *Requirements engineering* 18 (2013), número 1, páginas 25–41

[Ghosh et al. 2016] Ghosh, Shalini ; Elenius, Daniel ; Li, Wenchao ; Lincoln, Patrick ; Shankar, Natarajan ; Steiner, Wilfried: **"ARSENAL: automatic requirements specification extraction from natural language"**. En *NASA Formal Methods Symposium* Springer (evento), 2016, páginas 41–46

[Gokyer et al. 2008] Gokyer, Gokhan ; Cetin, Semih ; Sener, Cevat ; Yondem, Meltem T.: **"Non-functional requirements to architectural concerns: ML and NLP at crossroads"**. En *2008 the third international conference on software engineering advances* IEEE (evento), 2008, páginas 400–406

[Gramajo et al. 2020a] Gramajo, M ; Ballejos, L ; Ale, M: **"Seizing Requirements Engineering Issues through Supervised Learning Techniques"**. En *IEEE Latin America Transactions* 18 (2020), número 07, páginas 1164–1184

[Gramajo et al. 2018] Gramajo, María G. ; Ballejos, Luciana ; Ale, Mariel: **"Software requirements engineering through machine learning techniques: A literature review"**. En *2018 IEEE Biennial Congress of Argentina (ARGENCON)* IEEE (evento), 2018, páginas 1–7

[Gramajo et al. 2021] Gramajo, María G. ; Ballejos, Luciana ; Ale, Mariel: **"Recurrent Neural Networks to automate Quality assessment of Software Requirements"**. En *arXiv preprint arXiv:2105.04757* (2021)

[Gramajo et al. 2020b] Gramajo, María G. ; Ballejos, Luciana C. ; Ale, Mariel: **"Hacia la Evaluación Automática de la Calidad de los Requerimientos de Software usando Redes Neuronales Long Short Term Memory."**. En *WER*, 2020

[Halim y Siahaan 2019] Halim, Fahrizal ; Siahaan, Daniel: **"Detecting Non-Atomic Requirements in Software Requirements Specifications Using Classification Methods"**. En *2019 1st*

International Conference on Cybernetics and Intelligent System (ICORIS) Volumen 1 IEEE (evento), 2019, páginas 269–273

[Hayes et al. 2015] Hayes, Jane H. ; Li, Wenbin ; Yu, Tingting ; Han, Xue ; Hays, Mark ; Woodson, Clinton: **"Measuring requirement quality to predict testability"**. En *2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)* IEEE (evento), 2015, páginas 1–8

[Herbsleb y Moitra 2001] Herbsleb, James D. ; Moitra, Deependra: **"Global software development"**. En *IEEE software* 18 (2001), número 2, páginas 16–20

[Holtmann et al. 2011] Holtmann, Jörg ; Meyer, Jan ; Detten, Markus von: **"Automatic validation and correction of formalized, textual requirements"**. En *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops* IEEE (evento), 2011, páginas 486–495

[Hsia et al. 1993] Hsia, Pei ; Davis, Alan M. ; Kung, David C.: **"Status report: requirements engineering"**. En *IEEE software* 10 (1993), número 6, páginas 75–79

[Huertas y Juárez-Ramírez 2013] Huertas, Carlos ; Juárez-Ramírez, Reyes: **"Towards assessing the quality of functional requirements using English/Spanish controlled languages and context free grammar"**. En *Proc. Third International Conference on Digital Information and Communication Technology and its Applications (DICTAP 2013), Ostrava, Czech Republic on* Citeseer (evento), 2013, páginas 234–241

[Hussain et al. 2007] Hussain, Ishrar ; Ormandjieva, Olga ; Kosseim, Leila: **"Automatic quality assessment of SRS text by means of a decision-tree-based text classifier"**. En *Seventh International Conference on Quality Software (QSIC 2007)* IEEE (evento), 2007, páginas 209–218

[IEEE 1061 1998] IEEE 1061: **"IEEE Standard for a Software Quality Metrics Methodology"** / IEEE Computer Society. URL https://standards.ieee.org/ieee/1061/1549/, 1998. – Standard

[IEEE 15288 2014a] IEEE 15288: **"IEEE Standard for Application of Systems Engineering on Defense Programs"** / IEEE Computer Society. URL https://ieeexplore.ieee.org/document/7105318, 2014. – Standard

[IEEE 15288 2014b] IEEE 15288: **"IEEE Standard for Application of Systems Engineering on Defense Programs"** / IEEE Computer Society. URL https://ieeexplore.ieee.org/document/7105318, 2014. – Standard

[IEEE 830 1998] IEEE 830: **"IEEE Recommended Practice for Software Requirements Specifications"** / IEEE Computer Society. URL https://ieeexplore.ieee.org/document/720574, 1998. – Standard

[Ilieva y Ormandjieva 2005] Ilieva, MG ; Ormandjieva, Olga: **"Automatic transition of natural language software requirements specification into formal presentation"**. En *International Conference on Application of Natural Language to Information Systems* Springer (evento), 2005, páginas 392–397

[INCOSE 2017] INCOSE: **"Guide for Writing Requirements"**. En *International Council on Systems Engineering, San Diego/USA* (2017)

[ISO IEC IEEE 29148 2018] ISO IEC IEEE 29148: **"Systems and software engineering – Life cycle processes – Requirements engineering"** / International Organization for Standardization. URL https://www.iso.org/standard/72089.html, 2018. – Standard

[Jindal et al. 2016] Jindal, Rajni ; Malhotra, Ruchika ; Jain, Abha: **"Automated classification of security requirements"**. En *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* IEEE (evento), 2016, páginas 2027–2033

[Kaiya y Saeki 2006] Kaiya, Haruhiko ; Saeki, Motoshi: **"Using domain ontology as domain knowledge for requirements elicitation"**. En *14th IEEE International Requirements Engineering Conference (RE'06)* IEEE (evento), 2006, páginas 189–198

[Kalchbrenner et al. 2014] Kalchbrenner, Nal ; Grefenstette, Edward ; Blunsom, Phil: **"A Convolutional Neural Network for Modelling Sentences"**. En *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, páginas 655–665

[Kamata y Tamai 2007] Kamata, Mayumi I. ; Tamai, Tetsuo: **"How does requirements quality relate to project success or failure?"**. En *15th IEEE International Requirements Engineering Conference (RE 2007)* IEEE (evento), 2007, páginas 69–78

[Kamsties y Peach 2000] Kamsties, Erik ; Peach, Barbara: **"Taming ambiguity in natural language requirements"**. En *Proceedings of the Thirteenth international conference on Software and Systems Engineering and Applications* Volumen 1315, 2000

[Kenton y Toutanova 2019] Kenton, Jacob Devlin Ming-Wei C. ; Toutanova, Lee K.: **"Bert: Pre-training of deep bidirectional transformers for language understanding"**. En *Proceedings of naacL-HLT* Volumen 1, 2019, páginas 2

[Kitchenham y Pfleeger 1996] Kitchenham, Barbara ; Pfleeger, Shari L.: **"Software quality: the elusive target [special issues section]"**. En *IEEE software* 13 (1996), número 1, páginas 12–21

[Knauss et al. 2016] Knauss, Alessia ; Damian, Daniela ; Franch, Xavier ; Rook, Angela ; Müller, Hausi A. ; Thomo, Alex: **"ACon: A learning-based approach to deal with uncertainty in contextual requirements at runtime"**. En *Information and software technology* 70 (2016), páginas 85–99

[Knauss et al. 2009] Knauss, Eric ; Boustani, Christian E. ; Flohr, Thomas: **"Investigating the impact of software requirements specification quality on project success"**. En *International Conference on Product-Focused Software Process Improvement* Springer (evento), 2009, páginas 28–42

[Kocerka et al. 2018] Kocerka, Jerzy ; Krześlak, Michał ; Gałuszka, Adam: **"Analysing quality of textual requirements using natural language processing: A literature review"**. En *2018 23rd International Conference on Methods & Models in Automation & Robotics (MMAR)* IEEE (evento), 2018, páginas 876–880

[Konig et al. 2017] Konig, Fabio ; Ballejos, Luciana C. ; Ale, Mariel A.: **"A semi-automatic verification tool for software requirements specification documents"**. En *Simposio Argentino de Ingeniería de Software (ASSE)-JAIIO 46 (Córdoba, 2017).*, 2017

[Krogstie et al. 2013] KROGSTIE, John ; LINDLAND, Odd I. ; SINDRE, Guttorm: **"Towards a deeper understanding of quality in requirements engineering"**. En *Seminal Contributions to Information Systems Engineering*. Springer, 2013, páginas 89–102

[Kummler 2021] KUMMLER, Patrick: *Automated Quality Assessment of Natural Language Requirements*, Karlsruher Institut für Technologie (KIT), Tesis Doctoral, 2021

[Kurtanović y Maalej 2017] KURTANOVIĆ, Zijad ; MAALEJ, Walid: **"Automatically classifying functional and non-functional requirements using supervised machine learning"**. En *2017 IEEE 25th International Requirements Engineering Conference (RE)* Ieee (evento), 2017, páginas 490–495

[Lami et al. 2004] LAMI, Giuseppe ; GNESI, Stefania ; FABBRINI, Fabrizio ; FUSANI, Mario ; TRENTANNI, Gianluca: **"An automatic tool for the analysis of natural language requirements"**. En *Informe técnico, CNR Information Science and Technology Institute, Pisa, Italia, Setiembre* (2004)

[Lanubile et al. 1998] LANUBILE, Filippo ; SHULL, Forrest ; BASILI, Victor R.: **"Experimenting with error abstraction in requirements documents"**. En *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No. 98TB100262)* IEEE (evento), 1998, páginas 114–121

[Larman 2012] LARMAN, Craig: *Applying UML and patterns: an introduction to object oriented analysis and design and interative development*. Pearson Education India, 2012

[LeCun et al. 1998] LECUN, Yann ; BOTTOU, Léon ; BENGIO, Yoshua ; HAFFNER, Patrick: **"Gradient-Based Learning Applied to Document Recognition"**. En *Proceedings of the IEEE* 86 (1998), número 11, páginas 2278–2324

[Li 2017] LI, Tong: **"Identifying security requirements based on linguistic analysis and machine learning"**. En *2017 24th Asia-Pacific Software Engineering Conference (APSEC)* IEEE (evento), 2017, páginas 388–397

[Li et al. 2017] LI, Zeheng ; CHEN, Mingrui ; HUANG, LiGuo ; NG, Vincent ; GENG, Ruili: **"Tracing requirements in software design"**. En *Proceedings of the 2017 International Conference on Software and System Process*, 2017, páginas 25–29

[Lindgren et al. 2008] LINDGREN, Markus ; WALL, Anders ; LAND, Rikard ; NORSTRÖM, Christer: **"A method for balancing short-and long-term investments: Quality vs. Features"**. En *2008 34th Euromicro Conference Software Engineering and Advanced Applications* IEEE (evento), 2008, páginas 175–182

[Loucopoulos y Karakostas 1995] LOUCOPOULOS, Pericles ; KARAKOSTAS, Vassilios: *System requirements engineering*. McGraw-Hill, 1995

[Lv et al. 2021] LV, Jiaqi ; WU, Tianran ; PENG, Chenglun ; LIU, Yunpeng ; XU, Ning ; GENG, Xin: **"Compact learning for multi-label classification"**. En *Pattern Recognition* 113 (2021), páginas 107833

[Merten et al. 2016] MERTEN, Thorsten ; FALIS, Matúš ; HÜBNER, Paul ; QUIRCHMAYR, Thomas ; BÜRSNER, Simone ; PAECH, Barbara: **"Software feature request detection in issue tracking systems"**. En *2016 IEEE 24th International Requirements Engineering Conference (RE)* IEEE (evento), 2016, páginas 166–175

[Mezghani et al. 2018] MEZGHANI, Manel ; KANG, Juyeon ; SÈDES, Florence: **"Industrial requirements classification for redundancy and inconsistency detection in SEMIOS"**. En *2018 IEEE 26th International Requirements Engineering Conference (RE)* IEEE (evento), 2018, páginas 297–303

[Mich et al. 2004] MICH, Luisa ; FRANCH, Mariangela ; NOVI INVERARDI, Pierluigi: **"Market research for requirements analysis using linguistic tools"**. En *Requirements Engineering* 9 (2004), número 1, páginas 40–56

[Mikolov et al. 2010] MIKOLOV, Tomas ; KARAFIÁT, Martin ; BURGET, Lukas ; CERNOCKÝ, Jan ; KHUDANPUR, Sanjeev: **"Recurrent neural network based language model."**. En *Interspeech* Volumen 2 Makuhari (evento), 2010, páginas 1045–1048

[Mills y Haiduc 2017] MILLS, Chris ; HAIDUC, Sonia: **"A machine learning approach for determining the validity of traceability links"**. En *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* IEEE (evento), 2017, páginas 121–123

[Moreno et al. 2020] MORENO, Valentín ; GÉNOVA, Gonzalo ; PARRA, Eugenio ; FRAGA, Anabel: **"Application of machine learning techniques to the flexible assessment and improvement of requirements quality"**. En *Software Quality Journal* 28 (2020), número 4, páginas 1645–1674

[Moşteanu et al. 2020] MOŞTEANU, Dr ; ROXANA, Narcisa ; FACCIA, Dr ; CAVALIERE, Luigi Pio L. ; BHATIA, Saurav ; OTHERS: **"Digital technologies' implementation within financial and banking system during socio distancing restrictions–back to the future"**. En *International Journal of Advanced Research in Engineering and Technology* 11 (2020), número 6

[Navarro-Almanza et al. 2017] NAVARRO-ALMANZA, Raul ; JUAREZ-RAMIREZ, Reyes ; LICEA, Guillermo: **"Towards supporting software engineering using deep learning: A case of software requirements classification"**. En *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)* IEEE (evento), 2017, páginas 116–120

[Nikora y Balcom 2009] NIKORA, Allen P. ; BALCOM, Galen: **"Automated identification of LTL patterns in natural language requirements"**. En *2009 20th International Symposium on Software Reliability Engineering* IEEE (evento), 2009, páginas 185–194

[Ormandjieva et al. 2007] ORMANDJIEVA, Olga ; HUSSAIN, Ishrar ; KOSSEIM, Leila: **"Toward a text classification system for the quality assessment of software requirements written in natural language"**. En *Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting*, 2007, páginas 39–45

[Osman y Zaharin 2018] OSMAN, Mohd H. ; ZAHARIN, Mohd F.: **"Ambiguous software requirement specification detection: An automated approach"**. En *2018 IEEE/ACM 5th International Workshop on Requirements Engineering and Testing (RET)* IEEE (evento), 2018, páginas 33–40

[Ott 2013] OTT, Daniel: **"Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements"**. En *International Working Conference on Requirements Engineering: Foundation for Software Quality* Springer (evento), 2013, páginas 50–64

[Parra et al. 2015] Parra, Eugenio ; Dimou, Christos ; Llorens, Juan ; Moreno, Valentín ; Fraga, Anabel: **"A methodology for the classification of quality of requirements using machine learning techniques"**. En *Information and Software Technology* 67 (2015), páginas 180–195

[Pohl 2010] Pohl, Klaus: *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010

[Pressman y Maxim 2016] Pressman, Roger S. ; Maxim, Bruce R.: *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 2016

[Pudlitz et al. 2019] Pudlitz, Florian ; Brokhausen, Florian ; Vogelsang, Andreas: **"Extraction of system states from natural language requirements"**. En *2019 IEEE 27th International Requirements Engineering Conference (RE)* IEEE (evento), 2019, páginas 211–222

[Rahman et al. 2019] Rahman, Md A. ; Haque, Md A. ; Tawhid, Md Nurul A. ; Siddik, Md S.: **"Classifying non-functional requirements using RNN variants for quality software development"**. En *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, 2019, páginas 25–30

[Reenskaug 1979] Reenskaug, Trygve: **"Thing-Model-View-Editor: An Example from a Smalltalk-80 System"** / Xerox PARC. 1979. – Reporte de Investigació

[Saavedra 2016] Saavedra, R.: *Framework para la Verificación Semiautomática de Especificaciones de Requerimientos de Software*, Universidad Tecnológica Nacional, Tesis Maestría, 2016

[Saavedra et al. 2013] Saavedra, Roxana ; Ballejos, Luciana C. ; Ale, Mariel: **"Quality Properties Evaluation for Software Requirements Specifications: An Exploratory Analysis."**. En *ClbSE*, 2013, páginas 245–258

[del Sagrado y del Aguila 2018] Sagrado, José del ; Aguila, Isabel M. del: **"Stability prediction of the software requirements specification"**. En *Software Quality Journal* 26 (2018), número 2, páginas 585–605

[Sardinha et al. 2013] Sardinha, Alberto ; Chitchyan, Ruzanna ; Weston, Nathan ; Greenwood, Phil ; Rashid, Awais: **"EA-Analyzer: automating conflict detection in a large set of textual aspect-oriented requirements"**. En *Automated Software Engineering* 20 (2013), número 1, páginas 111–135

[Schneider et al. 1992] Schneider, G M. ; Martin, Johnny ; Tsai, Wei-Tek: **"An experimental study of fault detection in user requirements documents"**. En *ACM Transactions on Software Engineering and Methodology (TOSEM)* 1 (1992), número 2, páginas 188–204

[Seidl et al. 2015] Seidl, Martina ; Scholz, Marion ; Huemer, Christian ; Kappel, Gerti: *UML@ classroom*. Springer, 2015

[Sharma et al. 2014] Sharma, Richa ; Bhatia, Jaspreet ; Biswas, Kanad K.: **"Automated identification of business rules in requirements documents"**. En *2014 IEEE International Advance Computing Conference (IACC)* IEEE (evento), 2014, páginas 1442–1447

[Slankas y Williams 2013a] Slankas, John ; Williams, Laurie: **"Access control policy extraction from unconstrained natural language text"**. En *2013 International Conference on Social Computing* IEEE (evento), 2013, páginas 435–440

[Slankas y Williams 2013b] Slankas, John ; Williams, Laurie: **"Automated extraction of non-functional requirements in available documentation"**. En *2013 1st International workshop on natural language analysis in software engineering (NaturaLiSE)* IEEE (evento), 2013, páginas 9–16

[Sommerville 2011] Sommerville, Ian: *Software Engineering*. 9. Addison-Wesley, 2011

[Sommerville 2020] Sommerville, Ian: *Engineering software products*. Pearson London, 2020

[Swathi y Prasad 2011] Swathi, G. ; Prasad, Ch G.: **"Writing Software Requirements Specification Quality Requirements: An Approach to Manage Requirements Volatility"**. En *International Journal of Computer Technology and Applications* 2 (2011), número 3, páginas 631–638

[Verma y Kass 2008] Verma, Kunal ; Kass, Alex: **"Requirements analysis tool: A tool for automatically analyzing software requirements documents"**. En *International semantic web conference* Springer (evento), 2008, páginas 751–763

[Wagner 2013] Wagner, Stefan: *Software Product Quality Control*. Springer-Verlag Berlin Heidelberg, 2013 DOI: 10.1007/978-3-642-38571-1

[Wang 2016] Wang, Yinglin: **"Automatic semantic analysis of software requirements through machine learning and ontology approach"**. En *Journal of Shanghai Jiaotong University (Science)* 21 (2016), número 6, páginas 692–701

[Wiegers y Beatty 2013] Wiegers, Karl ; Beatty, Joy: *Software requirements*. Pearson Education, 2013

[Wiegers y Hokanson 2023] Wiegers, Karl ; Hokanson, Candase: *Software Requirements Essentials: Core Practices for Successful Business Analysis*. Pearson Education, 2023. – ISBN 9780138190286

[Wilson et al. 1997] Wilson, William M. ; Rosenberg, Linda H. ; Hyatt, Lawrence E.: **"Automated analysis of requirement specifications"**. En *Proceedings of the 19th international conference on Software engineering*, 1997, páginas 161–171

[Winkler y Vogelsang 2016] Winkler, Jonas ; Vogelsang, Andreas: **"Automatic classification of requirements based on convolutional neural networks"**. En *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)* IEEE (evento), 2016, páginas 39–45

[Yang et al. 2010a] Yang, Hui ; De Roeck, Anne ; Gervasi, Vincenzo ; Willis, Alistair ; Nuseibeh, Bashar: **"Extending nocuous ambiguity analysis for anaphora in natural language requirements"**. En *2010 18th IEEE International Requirements Engineering Conference* IEEE (evento), 2010, páginas 25–34

[Yang et al. 2011] Yang, Hui ; De Roeck, Anne ; Gervasi, Vincenzo ; Willis, Alistair ; Nuseibeh, Bashar: **"Analysing anaphoric ambiguity in natural language requirements"**. En *Requirements engineering* 16 (2011), número 3, páginas 163–189

[Yang et al. 2012] YANG, Hui ; DE ROECK, Anne ; GERVASI, Vincenzo ; WILLIS, Alistair ; NUSEIBEH, Bashar: **"Speculative requirements: Automatic detection of uncertainty in natural language requirements"**. En *2012 20th IEEE International Requirements Engineering Conference (RE)* IEEE (evento), 2012, páginas 11–20

[Yang et al. 2010b] YANG, Hui ; WILLIS, Alistair ; DE ROECK, Anne ; NUSEIBEH, Bashar: **"Automatic detection of nocuous coordination ambiguities in natural language requirements"**. En *Proceedings of the IEEE/ACM international conference on Automated software engineering*, 2010, páginas 53–62

[Young et al. 2018] YOUNG, Tom ; HAZARIKA, Devamanyu ; PORIA, Soujanya ; CAMBRIA, Erik: **"Recent trends in deep learning based natural language processing"**. En *ieee Computational intelligenCe magazine* 13 (2018), número 3, páginas 55–75

Lista de Figuras

Lista de Tablas

I

Pseudocódigos

.1/ Algoritmo para Preprocesamiento de Requerimientos

Data: DataFrame df with a $Requirements$ column
Result: DataFrame df with additional $Tokenized$, Pos_tagged, and $Tagged\_Req$ columns. The DataFrame is also saved to an excel file
Initialize an empty list 'token_list';
for each requirement in 'Requirements' column of 'df' **do**
    Tokenize the requirement text into individual words;
    Add tokens to 'token_list';
end
Add 'token_list' as a new column 'Tokenized' in 'df';
Initialize an empty list 'pos_tagged_list';
for each token in 'Tokenized' column of 'df' **do**
    Assign part of speech tags to the token;
    Add tagged tokens to 'pos_tagged_list';
end
Add 'pos_tagged_list' as a new column 'Pos_tagged' in 'df';
Initialize an empty list 'tagged_sentences_list';
for each tagged_sentence in 'Pos_tagged' column of 'df' **do**
    Initialize an empty list 'tagged_sentence_list';
    for each word in tagged_sentence **do**
        Add tag of word to 'tagged_sentence_list';
    end
    Add 'tagged_sentence_list' to 'tagged_sentences_list';
end
Join each 'tagged_sentence_list' in 'tagged_sentences_list' by ',';
Add joined 'tagged_sentences_list' as a new column 'Tagged_Req' in 'df';
Save 'df' to an Excel file;

Algoritmo 1: Preprocesamiento de requerimientos.

El algoritmo presentado muestra un proceso que consta de varias etapas para llevar a cabo el preprocesamiento y etiquetado de datos textuales, tomados de una colección de requerimientos, contenidos en un objeto DataFrame df.

La primera etapa de este proceso describe la tokenización, que consiste en desglosar el texto en palabras individuales o tokens. Esto se logra utilizando la función *word_tokenize* del Natural Language Toolkit (NLTK). La lista resultante de tokens se agrega a *token_list*, y el proceso completo se repite para cada entrada en la columna Requirements del DataFrame. Una vez que todos los requerimientos han sido tokenizados, *token_list* se agrega como una nueva columna al DataFrame, etiquetada como 'Tokenized'.

La segunda etapa del proceso implica etiquetar las partes de la oración, un procedimiento que asigna a cada token una etiqueta gramatical (sustantivo, verbo, adjetivo, etc.). Esto se realiza utilizando la función *pos_tag*, también de la biblioteca NLTK, que toma los requerimientos tokenizados como entrada. La salida es una lista anidada, donde cada requerimiento se representa como una lista de tuplas. Cada tupla contiene un token y su etiqueta correspondiente. Esta lista etiquetada con las partes de la oración se agrega como otra nueva columna al DataFrame, etiquetada como *'Pos_tagged'*.

En la etapa final, se extraen las etiquetas de las partes de la oración para cada requerimiento y se concatenan en una sola cadena, separadas por comas. Esto se hace iterando a través de cada requerimiento etiquetado, extrayendo la etiqueta de cada tupla (ignorando el token), y uniendo las etiquetas. Esta operación resulta en una lista de cadenas, que se agrega al DataFrame como una última nueva columna, etiquetada como *'Tagged_Req'*. A partir de esto se puede obtener un vector con las etiquetas que representan cada palabra de un requerimiento.

Finalmente, el DataFrame actualizado, que incluye los requerimientos originales, su forma tokenizada, los tokens etiquetados con las partes de la oración, y las etiquetas separadas, se guarda en un archivo de Excel para su uso y análisis futuro. Este procedimiento transforma efectivamente los requerimientos textuales brutos en una forma más estructurada que sirve como entrada para los modelos de aprendizaje automático profundo.

.2/ Algoritmo para Definición y Entrenamiento de Modelos Neuronales

Data: DataFrame df with a $Requirements$ and $Tagged_Req$ column
Result: Neural Network Model Trained and Optimazed
Import required external libraries and modules ;
Define the RequirementRNN class ;
Separate the dataset based on certain conditions and balance it using sampling ;
Tokenize and tag the 'Tagged_Req' data ;
Transform the input data using TextFeaturizer and Padder2d ;
Set up the neural network model (RequirementRNN) using the NeuralNetClassifier from skorch ;
Set up the optimizer ;
Define a set of hyperparameters for the model: embedding_dim, rec_layer_type, num_units, num_layers, dropout, learning_rate, and max_epochs;
Define a set of evaluation metrics: accuracy, recall, precision, f1 score, and jaccard similarity;
Implement RandomizedSearchCV for hyperparameter tuning, using the neural network model, parameters, number of iterations, verbosity, refit criterion, evaluation metrics, and number of folds for cross-validation;
Get the best score and best parameters from the RandomizedSearchCV;
Save the model generated ;

Algoritmo 2: Definición y Entrenamiento de Modelos Neuronales.

.3/ Algoritmo para Long-Short Term Memory

Data: Matrix compose of a Sequence of tokens of a fixed length. One token per word in a requirement, each row represent a requirement.

Result: An "output" tridimensional tensor, which contains the output features (h_t) from the last layer of the LSTM for each time step.

for each timestep t in the input sequence **do**

1. Compute the forget gate output:
- $forget\_gate_t = sigmoid(W_f * [h_{(t-1)}, x_t] + b_f)$;
2. Compute the input gate outputs:
- $input\_gate_t = sigmoid(W_i * [h_{(t-1)}, x_t] + b_i)$;
- $candidate\_gate_t = tanh(W_C * [h_{(t-1)}, x_t] + b_C)$;
3. Update the cell state:
- $C_t = forget\_gate_t * C_{(t-1)} + input\_gate_t * candidate\_gate_t$;
4. Compute the output gate output:
- $output\_gate_t = sigmoid(W_o * [h_(t-1), x_t] + b_o)$;
5. Update the hidden state:
- $h_t = output\_gate_t * tanh(C_t)$;

end

Algoritmo 3: Long-Short Term Memory.

.4/ Algoritmo para Gated Recurrent Unit

Data: Matrix compose of a Sequence of tokens of a fixed length. One token per word in a requirement, each row represent a requirement.
Result: An "output" tridimensional tensor, which contains the output features (h_t) from the last layer of the GRU for each time step.
for each timestep t in the input sequence **do**
1. Compute the forget gate output:
- $forget\_gate_t = sigmoid(W_u * [h_{(t-1)}, x_t] + b_u)$;
2. Compute the reset gate output::
- $reset\_gate_t = sigmoid(W_r * [h_{(t-1)}, x_t] + b_r)$;
3. Compute the candidate hidden state:
- $candidate_{h_t} = tanh(W * [reset\_gate_t * h_{(t-1)}, x_t] + b)$;
4. Update the hidden state:
- $h_t = (1 - update\_gate_t) * h_{(t-1)} + update\_gate_t * candidate_{h_t}$;

end

Algoritmo 4: Gated Recurrent Unit.

En los pseudocódigos anteriores, se utilizan las siguientes operaciones:

- $*$ represents elementwise multiplication,
- $[a, b]$ represents the concatenation of a and b,
- W_* and b_* are the weights and bias parameters for the corresponding gates,
- h_t is the hidden state at time t,
- x_t is the input at time t,
- C_t is the cell state at time t for LSTM,
- sigmoid and tanh are the sigmoid and hyperbolic tangent activation functions.

.5/ Algoritmo para Bidirectional Encoder Representations from Transformers

1. *Convertir cada token a un vector de incrustación utilizando la tokenización de WordPiece*: WordPiece es una forma de tokenización que permite dividir palabras en subpalabras, lo que puede ser útil para lidiar con palabras fuera del vocabulario. Después de tokenizar una secuencia de entrada en tokens, cada token se convierte en un vector de incrustación, que es una representación de alta dimensión del token que puede ser procesada por la red neuronal.
2. *Agregar codificación posicional a la incrustación*: Como los Transformers no tienen ninguna noción inherente del orden de los tokens en una secuencia, se agrega información posicional a las incrustaciones. Esta información permite que la red entienda el orden de los tokens en la secuencia.
3. *Para cada capa en BERT*: BERT se compone de múltiples capas idénticas, cada una de las cuales procesa las incrustaciones de los tokens y las pasa a la siguiente capa.

4. *Para cada token en la secuencia de entrada*: En cada capa, cada token se procesa por separado.
5. *Calcular Consulta (Query), Clave (Key) y Valor (Value)*: Cada token se transforma en tres vectores diferentes: un vector de consulta, uno de clave y uno de valor. Estos vectores se utilizan en el mecanismo de atención para determinar cuánta "atención" debe prestar cada token a cada otro token.
6. *Calcular puntuaciones de atención*: La atención se calcula tomando el producto escalar de los vectores de consulta y clave, dividiéndolo por la raíz cuadrada de la dimensión de los vectores y aplicando la función softmax para obtener una distribución de probabilidad.
7. *Calcular la suma ponderada de los valores*: Las puntuaciones de atención se utilizan para ponderar los vectores de valor. Esto significa que cada token tiene una representación que es una suma de los vectores de valor de todos los demás tokens, donde los vectores se ponderan por cuánta "atención" el token debe prestar a cada otro token.
8. *Pasar la suma ponderada a través de una red neuronal de avance (feed-forward) y normalizar*: La suma ponderada se pasa a través de una red neuronal de avance. El resultado se suma a la representación original del token (antes de la atención y la red de avance) y se normaliza.
9. *Salida*: La salida de la última capa se utiliza como la representación final de cada token en la secuencia.

Variables:

- W_q, W_k, W_v: Son las matrices de pesos utilizadas para transformar el vector de entrada en los vectores de consulta, clave y valor, respectivamente.
- X: Representa las incrustaciones de entrada para cada token.
- d_k: Es la dimensión de los vectores de consulta/clave.
- Q, K, V: Son los vectores de consulta, clave y valor, respectivamente.
- $scores$: Son las puntuaciones de atención.
- Z: Es la suma ponderada de los vectores de valor.
- FFN_{output}: Es la salida de la red neuronal de avance.
- $Layer_{output}$: Es la salida final de cada capa después de la normalización.

Data: Matrix compose of a Sequence of tokens of a fixed length. One token per word in a requirement, each row represent a requirement. Additionally to that one a Segment ID that enumerate the sentence in each requirement and a Positional Encodings.

Result: An "output" tridimensional tensor, which contains the output features (h_t) from the last layer of the BERT for each time step.

for each token in the input requirement **do**
    Convert token to embedding vector using WordPiece tokenization ;
    Add positional encoding to the embedding ;
end
for each layer in BERT **do**
    for each token in the input requirement **do**
        1. Compute Query:
        $Q = W_q * X$;
        2. Compute Key:
        $K = W_k * X$;
        3. Compute Value:
        $V = W_v * X$;
        4. Compute attention scores:
        $scores = \text{softmax}((Q * K^T)/\sqrt{d_k})$;
        5. Compute weighted sum of the values:
        $Z = scores * V$;
        6. Pass Z through a feed-forward neural network:
        $FFN_{output} = FFN(Z)$;
        7. Add the input to the layer (X) to the output of the feedforward network and apply layer normalization: $Layer_{output} = LayerNorm(X + FFN_{output})$;
    end
end

Algoritmo 5: Bidirectional Encoder Representations from Transformers

II

Modelos Neuronales Resultantes

.1/ Modelos con Redes Neuronales Recurrentes

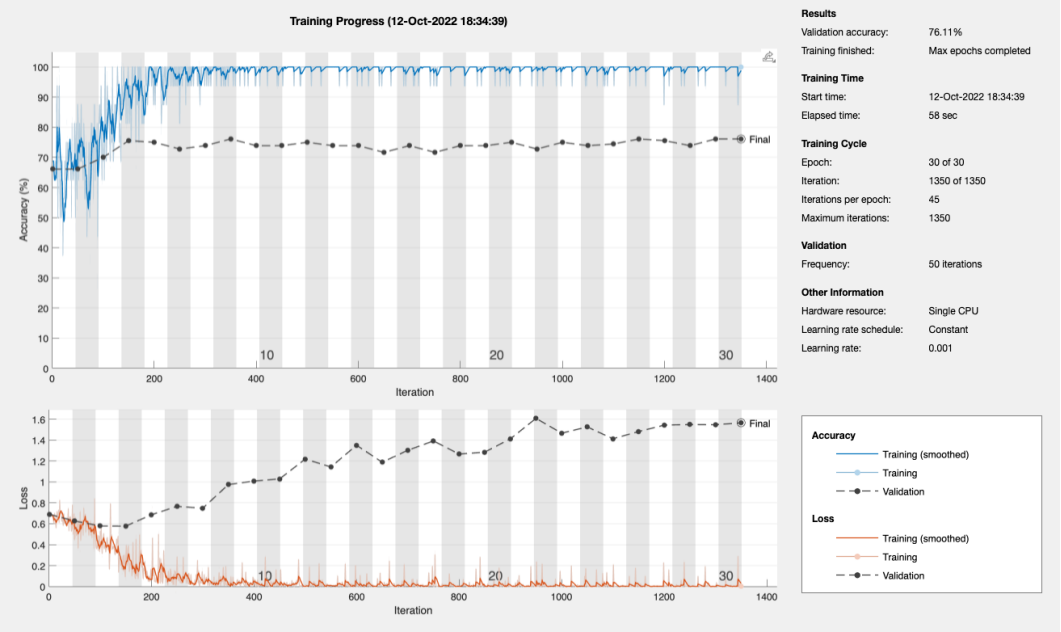


Figura 1: Característica de Calidad Ambiguo.

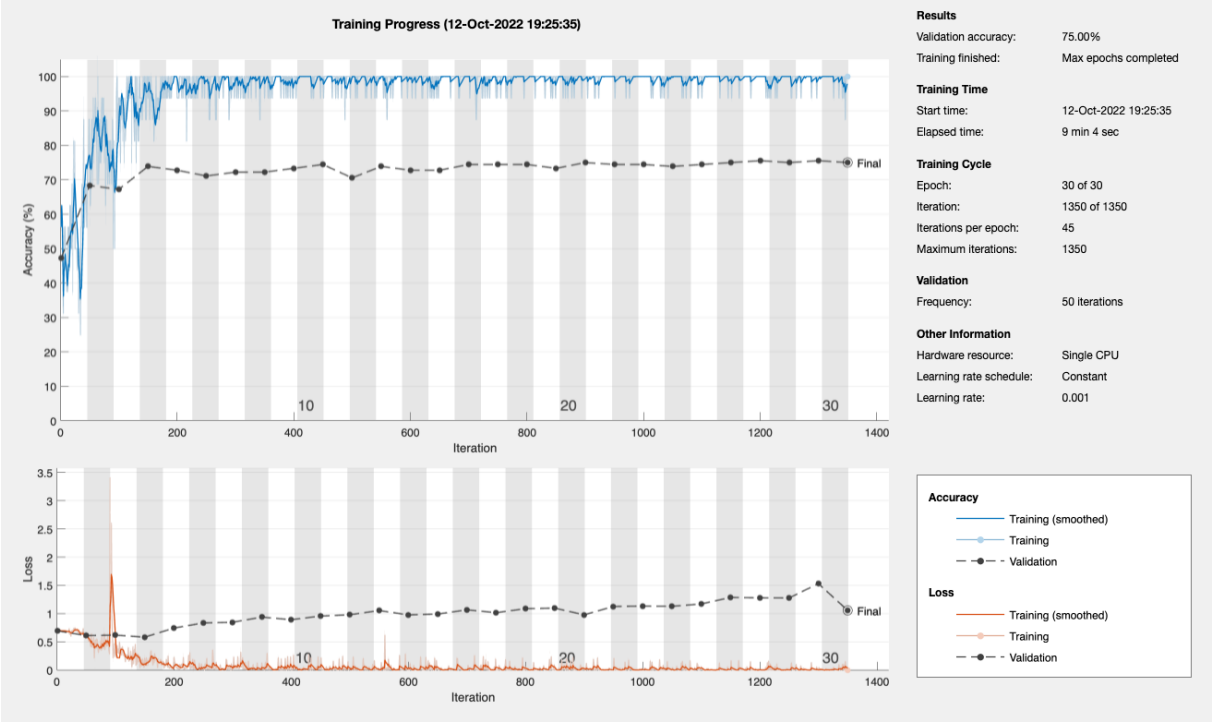


Figura 2: Característica de Calidad Conforme.

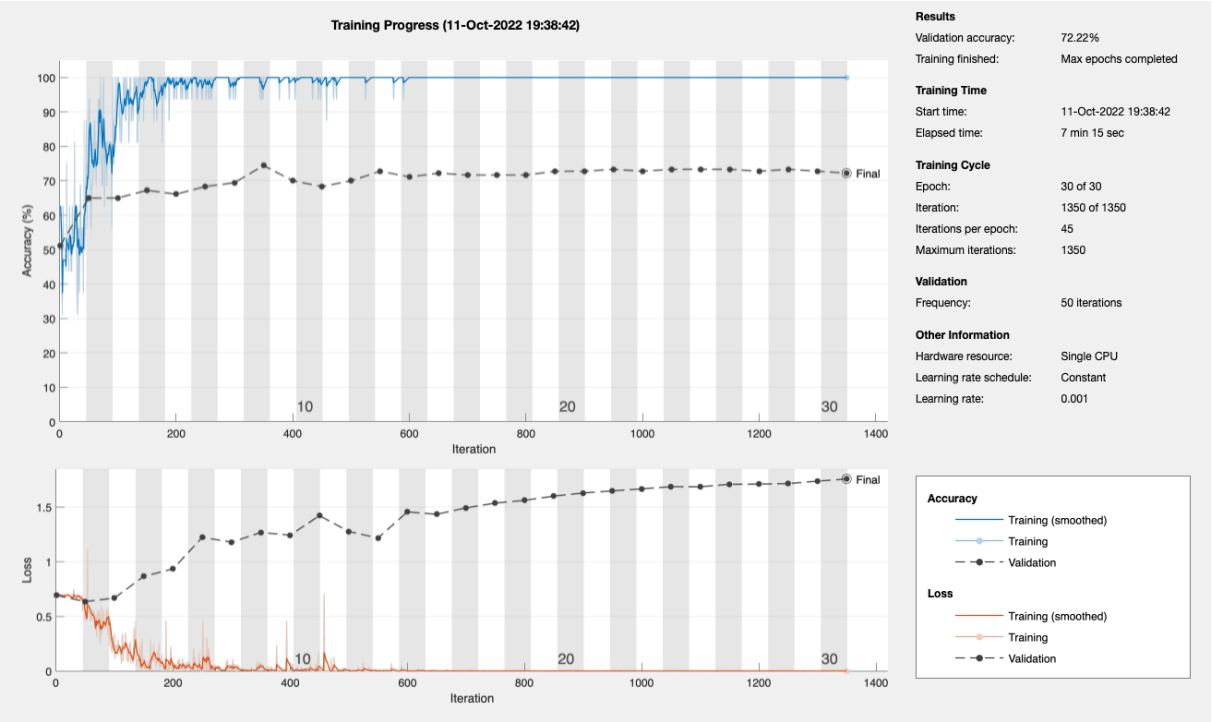


Figura 3: Característica de Calidad Correcto.

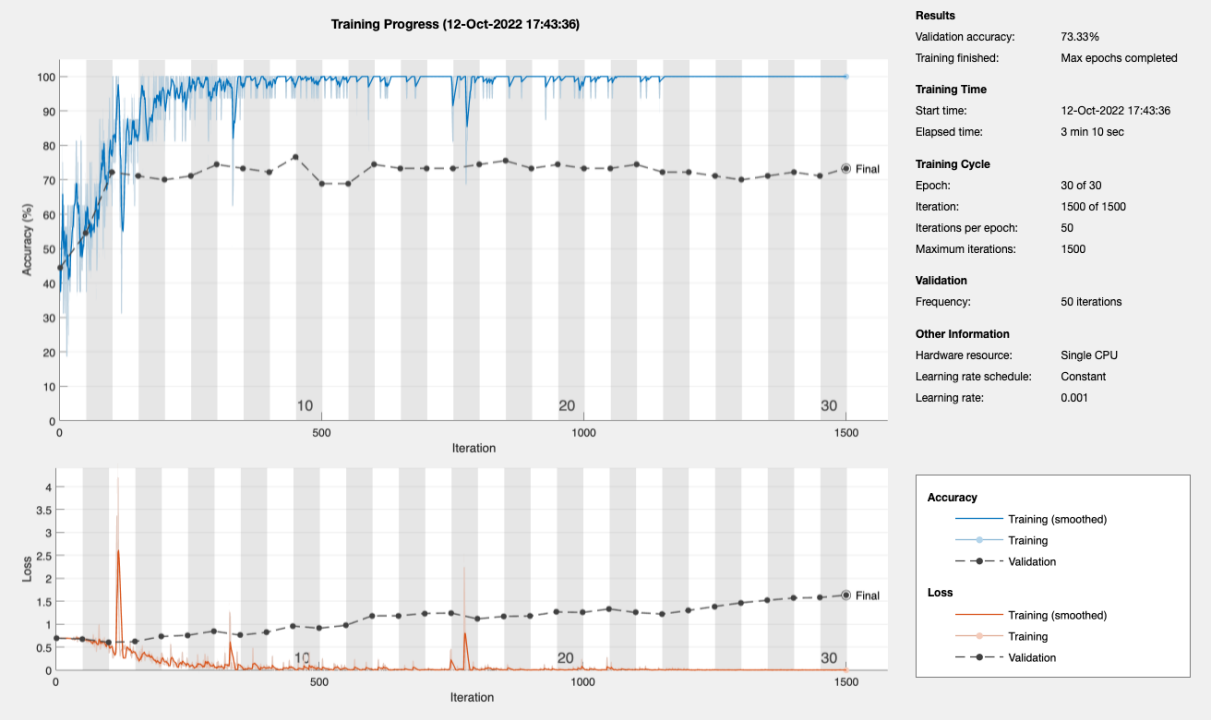


Figura 4: Característica de Calidad Factible.

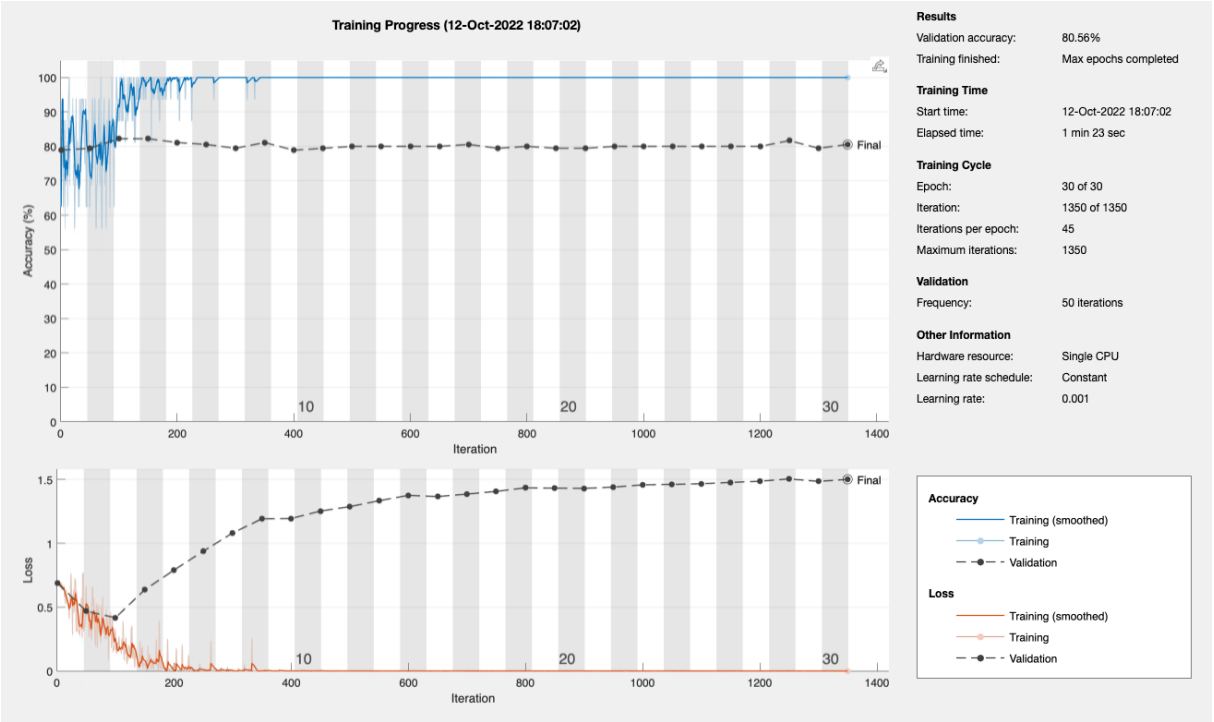


Figura 5: Característica de Calidad Singular.

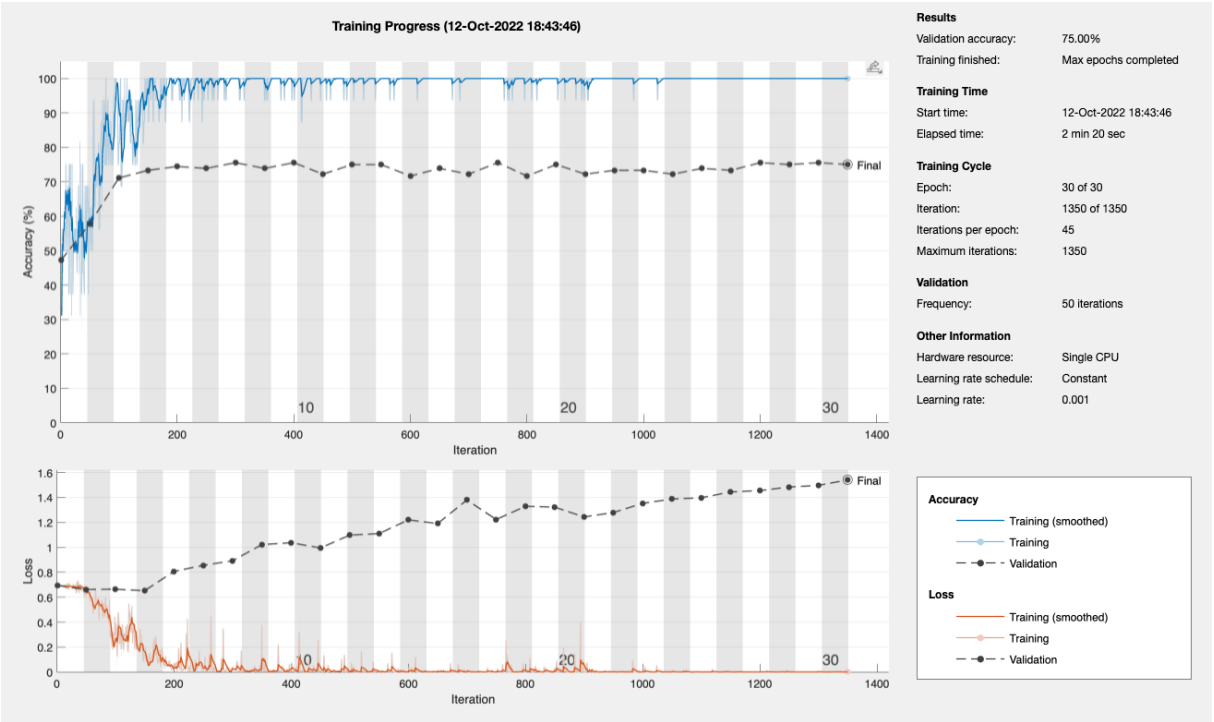


Figura 6: Característica de Calidad Verificable.

.2/ Modelos con Redes Neuronales Transformadoras

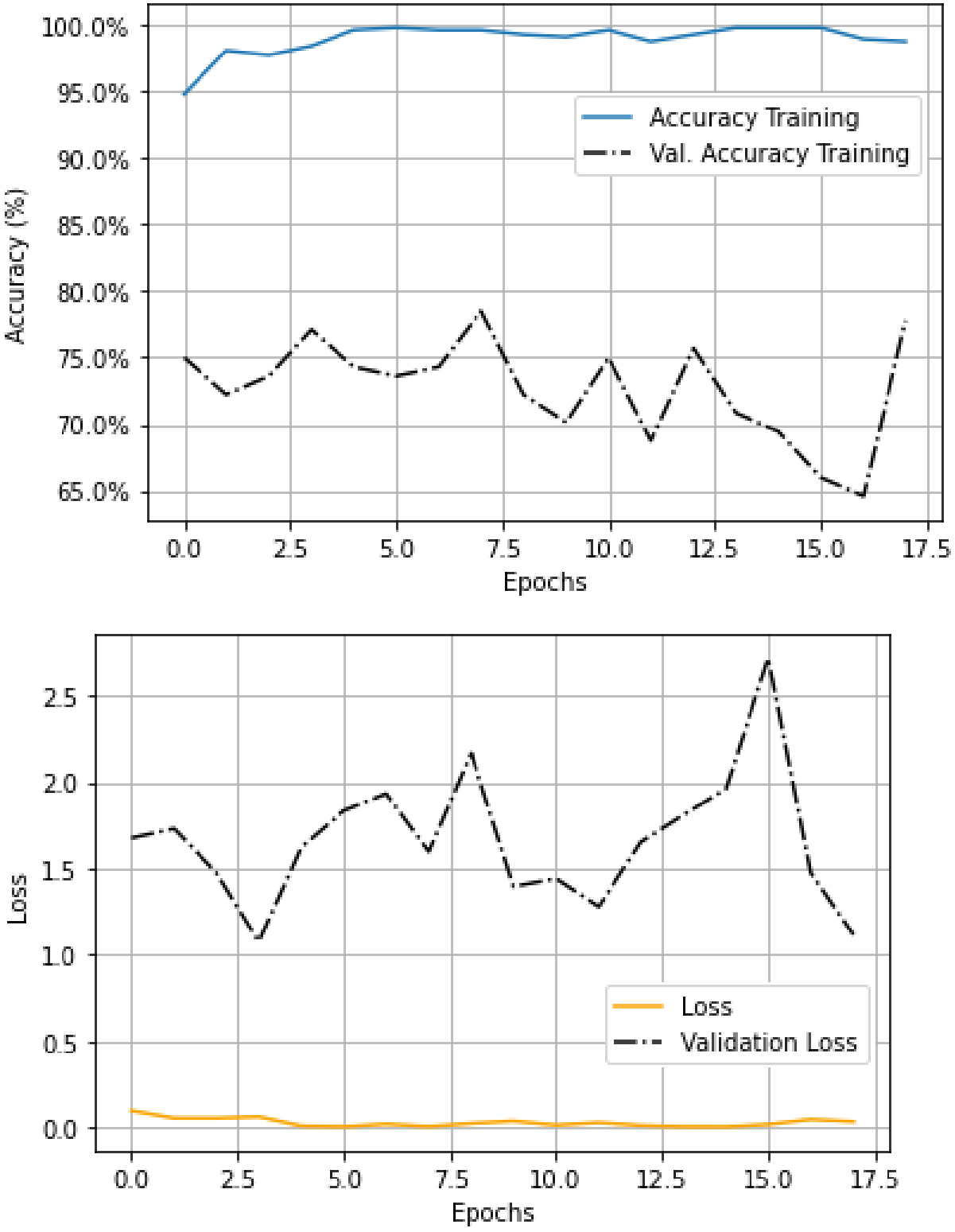


Figura 7: Característica de Calidad Ambiguo con BERT.

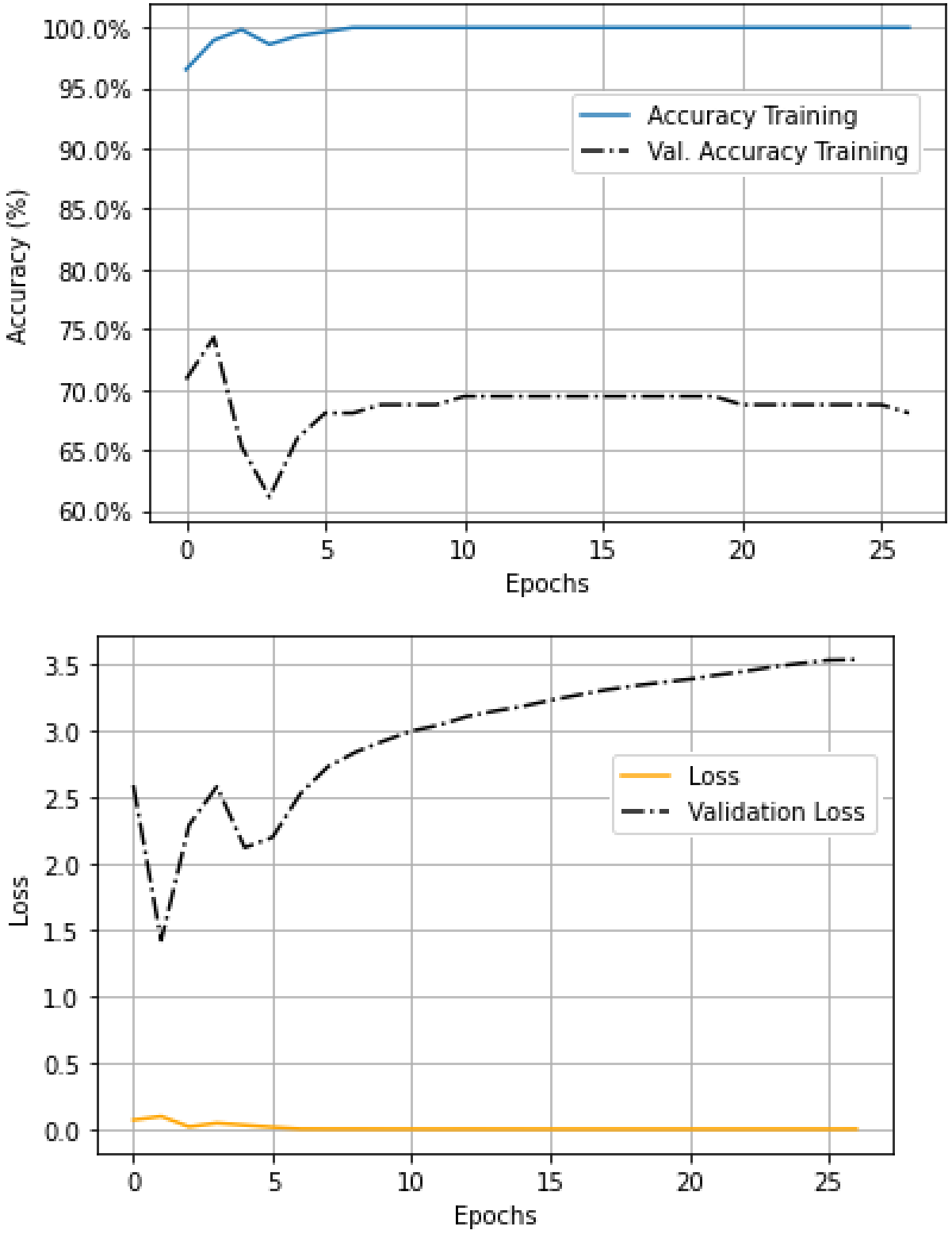


Figura 8: Característica de Calidad Conforme con BERT.

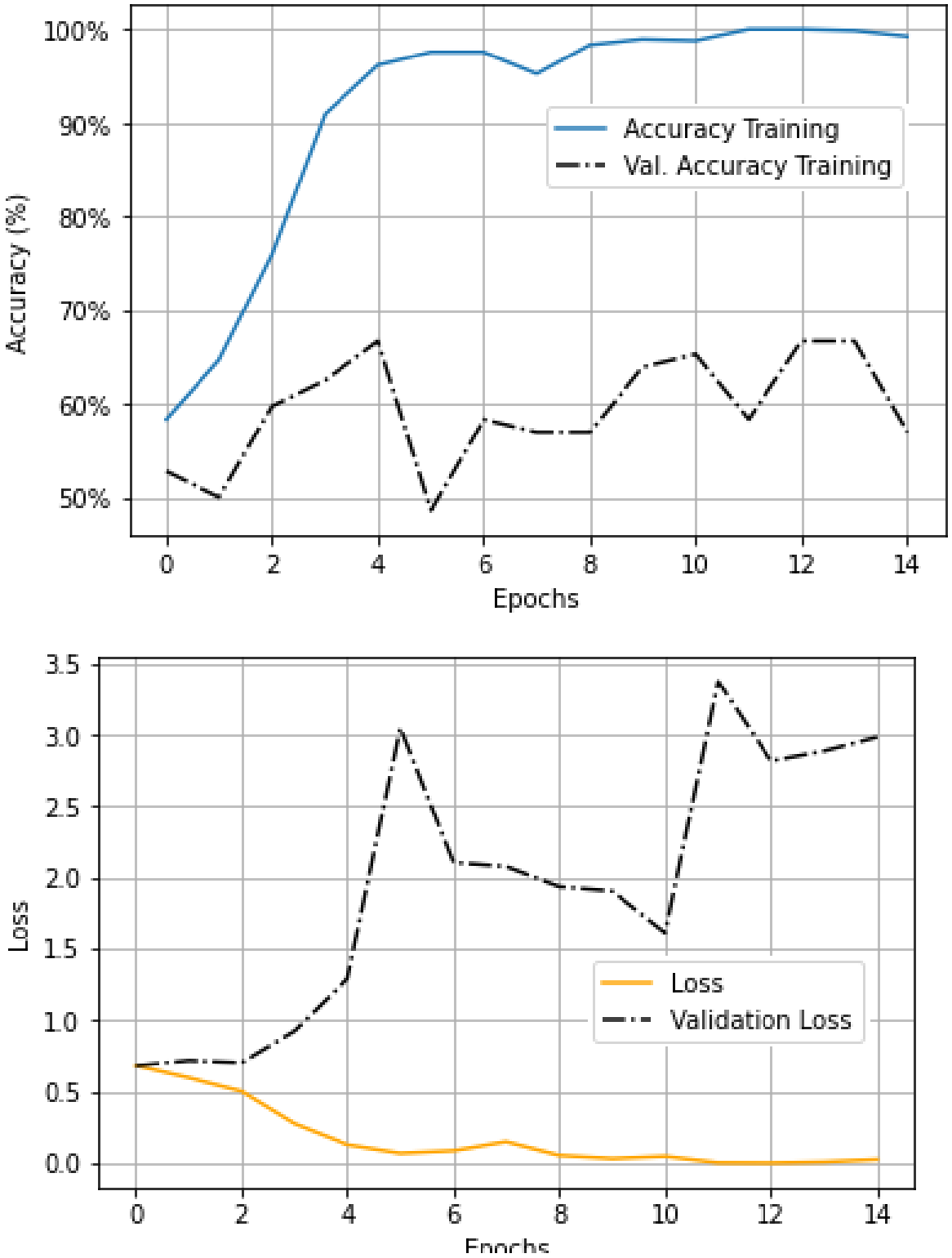


Figura 9: Característica de Calidad Correcto con BERT.

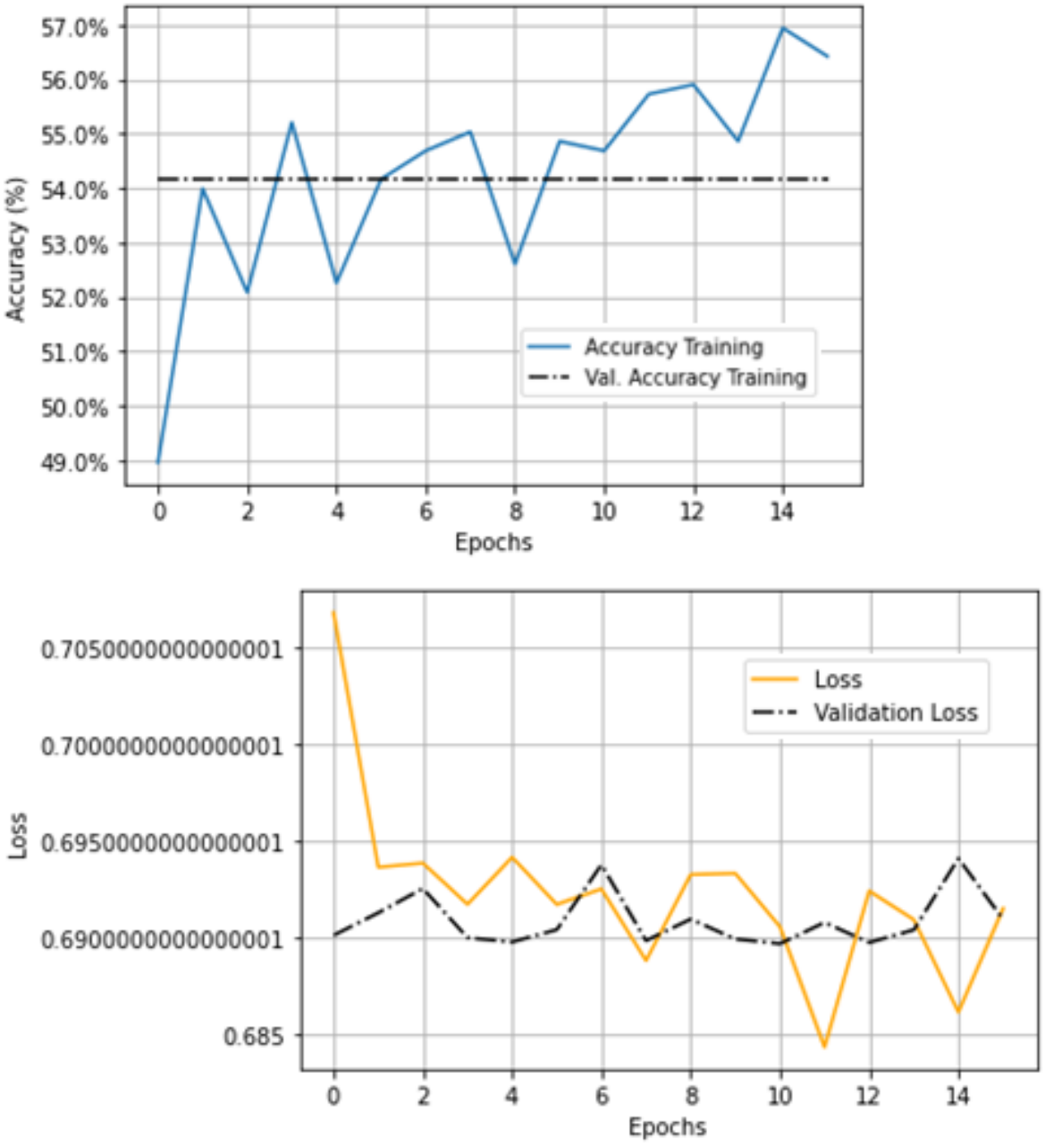


Figura 10: Característica de Calidad Factible con BERT.

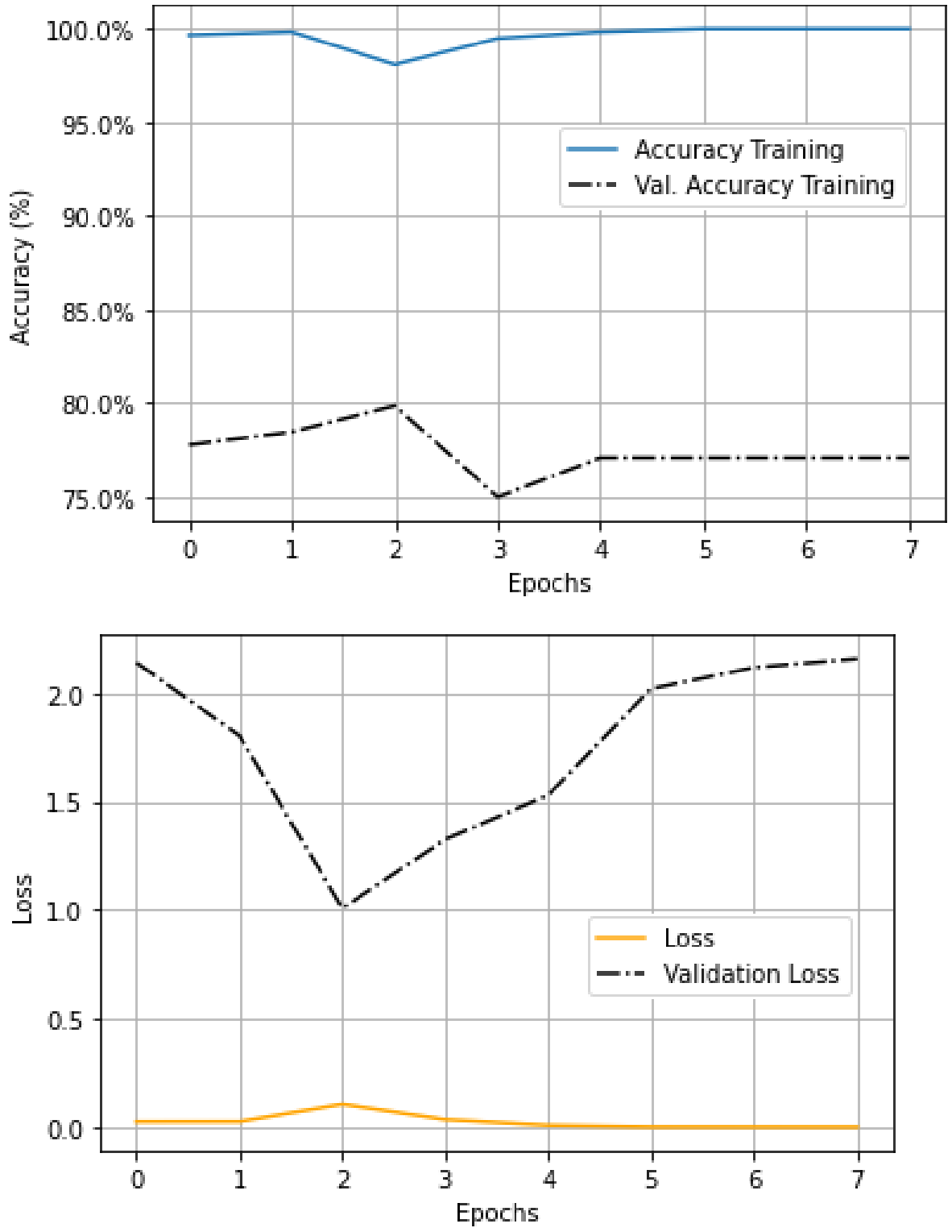


Figura 11: Característica de Calidad Singular con BERT.

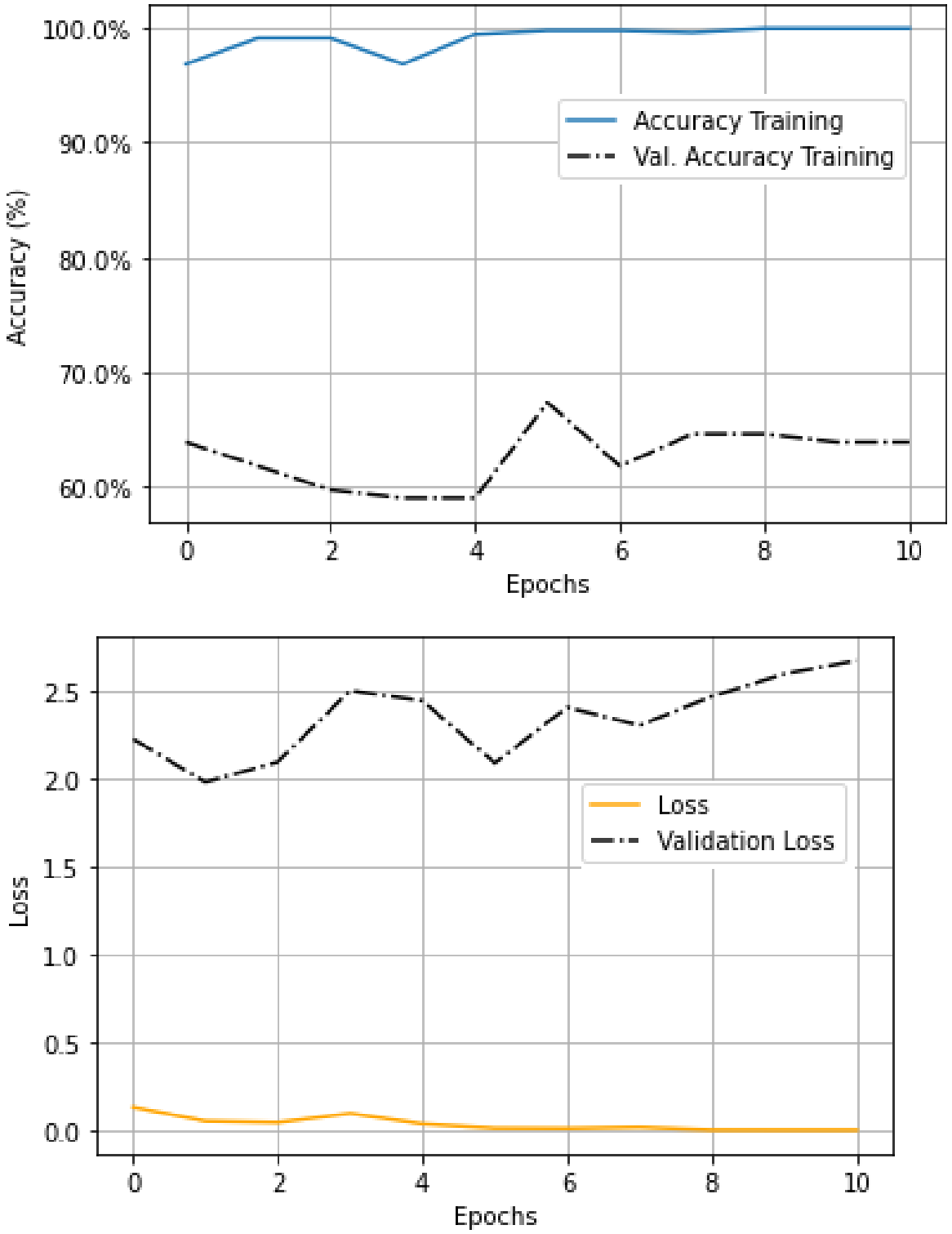


Figura 12: Característica de Calidad Verificable con BERT.

III

Casos de Uso

Tabla 1: Descripción caso de uso Cargar Lote de Requerimientos.

Acción del actor	**Respuesta del sistema**
1. Este caso de uso comienza cuando el Ingeniero de Requerimientos desea ingresar un conjunto de requerimientos para su evaluación.	
2. El Ingeniero de Requerimientos presiona el botón "Upload File" y selecciona el archivo que desea cargar y presiona el botón "Continue".	**3.** El sistema valida el formato del archivo, carga en la base de datos y presenta los requerimientos contenidos en el mismo.
4. El Ingeniero de Requerimientos visualiza los requerimientos interpretados por el sistema.	
Curso Alternativo	
Acción del actor	
2.a El formato de archivo no es válido.	El sistema muestra un mensaje de error "El archivo no es válido."

Un caso de uso es un documento narrativo que describe la secuencia de eventos de un actor que utiliza un sistema para completar un proceso. A continuación, se describen los casos de uso que comprende el despliegue de la primera versión de la herramienta QASS.

.0.1/ Caso de Uso 01 - Cargar Lote de Requerimientos

Actor principal: Ingeniero de Requerimientos.

Precondiciones: El Ingeniero de Requerimientos debe estar previamente registrado en el sistema para iniciar la carga de requerimientos.

Postcondiciones: Los requerimientos de entrada se cargan correctamente en el sistema.

Escenario principal de éxito: Este caso de uso comienza cuando el Ingeniero de Requerimientos desea cargar un conjunto de requerimientos de entrada para realizar la evaluación de calidad de estos últimos. El sistema valida el formato del archivo provisto y permite su carga.

La Tabla 1 presenta la descripción del caso de uso correspondiente a Cargar Lote de Requerimientos.

Cabe mencionar que UML permite agrupar aquellos casos de uso que comparten el mismo objetivo, en esta caso particular, realizar ajustes sobre los requerimientos. Los pasos individuales correspondientes a cada caso de uso deben detallarse en la descripción del curso normal de los eventos Seidl et al. (2015). Específicamente, Ajustar Lote de Requerimientos agrupa los casos de uso, Añadir, Editar y Eliminar Requerimiento. Estos últimos, se describen a continuación.

.0.2/ Caso de Uso 02 A - Añadir Requerimiento

Actor principal: Ingeniero de Requerimientos.

Precondiciones: El Ingeniero de Requerimientos debe estar registrado en el sistema para añadir un requerimiento. El archivo con los requerimientos de entrada debe estar previamente cargado.

Tabla 2: Descripción caso de uso Añadir Requerimiento.

Acción del actor	**Respuesta del sistema**
1. Este caso de uso comienza cuando el Ingeniero de Requerimientos desea añadir un nuevo requerimiento.	
2. El Ingeniero de Requerimientos presiona el botón "New Requirement".	**3.** El sistema muestra el campo descripción.
4.El Ingeniero de Requerimientos ingresa la descripción del nuevo requerimiento y presiona el botón "Save".	**5.** El sistema registra el nuevo requerimiento correctamente.
6. El Ingeniero de Requerimientos presiona el botón "Continue".	**7.** El sistema muestra la página correspondiente a Evaluar Requerimientos.

Postcondiciones: El sistema añade el nuevo requerimiento correctamente.

Escenario principal de éxito: Este caso de uso comienza cuando el Ingeniero de Requerimientos desea ingresar un nuevo requerimiento. Para ello, ingresa la descripción del requerimiento. El sistema permite añadir el requerimiento correctamente.

La Tabla 2 presenta la descripción del caso de uso correspondiente a Añadir Requerimiento.

.0.3/ Caso de Uso 02 B - Editar Requerimiento

Actor principal: Ingeniero de Requerimientos.

Precondiciones: El Ingeniero de Requerimientos debe estar registrado en el sistema para editar un requerimiento. El requerimiento que se desea editar debe estar previamente cargado en el sistema.

Postcondiciones: El sistema registra la edición del requerimiento correctamente.

Escenario principal de éxito: Este caso de uso comienza cuando el Ingeniero de Requerimientos desea editar un requerimiento. Para ello, selecciona el requerimiento que desea editar y realiza modificaciones sobre su descripción. El sistema registra la edición de requerimiento de forma correcta.

La Tabla 3 presenta la descripción del caso de uso correspondiente a Editar Requerimiento.

.0.4/ Caso de Uso 02 C - Eliminar Requerimiento

Actor principal: Ingeniero de Requerimientos.

Precondiciones: El Ingeniero de Requerimientos debe estar registrado en el sistema para eliminar un requerimiento. El requerimiento que se desea eliminar debe estar previamente cargado en el sistema.

Postcondiciones: El sistema elimina el requerimiento correctamente.

Escenario principal de éxito: Este caso de uso comienza cuando el Ingeniero de Requerimientos desea eliminar un requerimiento. Para ello, selecciona el requerimiento que se desea eliminar. El sistema registra la eliminación el requerimiento de forma correcta.

La Tabla 4 presenta la descripción del caso de uso correspondiente a Eliminar Requerimiento.

Tabla 3: Descripción caso de uso Editar Requerimiento.

Acción del actor	**Respuesta del sistema**
1. Este caso de uso comienza cuando el Ingeniero de Requerimientos desea editar un requerimiento.	
2. El Ingeniero de Requerimientos selecciona el requerimiento que desea modificar y presiona el botón "Edit".	**3.** El sistema muestra el campo descripción.
4.El Ingeniero de Requerimientos realiza la edición del requerimiento en el campo descripción y presiona el botón "Save".	**5.** El sistema registra la edición del requerimiento.
6. El Ingeniero de Requerimientos presiona el botón "Continue".	**7.** El sistema muestra la página correspondiente a Evaluar Requerimientos.

.0.5/ Caso de Uso 03 - Evaluar Requerimientos

Actor principal: Ingeniero de Requerimientos.

Precondiciones: El Ingeniero de Requerimientos debe estar registrado en el sistema para evaluar los requerimientos a través del servicio de IA. Los requerimientos de entrada deben estar previamente cargados.

Postcondiciones: El sistema mediante el servicio de IA integrado evalúa los requerimientos correctamente.

Escenario principal de éxito: Este caso de uso comienza cuando el Ingeniero de Requerimientos desea procesar los requerimientos cargados para evaluar su calidad. Para ello, selecciona la opción ejecutar servicio de IA con configuración predeterminada. El sistema evalúa los requerimientos.

La Tabla 5 muestra la descripción del caso de uso correspondiente a Evaluar Requerimiento.

.0.6/ Caso de Uso 04 - Verificar Requerimientos

Actor principal: Ingeniero de Requerimientos.

Precondiciones: El Ingeniero de Requerimientos debe estar registrado en el sistema para visualizar los resultados. El servicio de IA debe ser ejecutado previamente.

Postcondiciones: El sistema presenta los resultados correctamente.

Escenario principal de éxito: Este caso de uso comienza cuando el Ingeniero de Requerimientos desea verificar los resultados de la ejecución del servicio de IA. Para ello, selecciona la opción visualizar resultados. El sistema muestra un reporte con los resultados obtenidos de la ejecución del servicio de IA.

La Tabla 6 muestra la descripción del caso de uso correspondiente a Verificar Requerimiento.

Tabla 4: Descripción caso de uso Eliminar Requerimiento.

Acción del actor	**Respuesta del sistema**
1. Este caso de uso comienza cuando el Ingeniero de Requerimientos desea eliminar un requerimiento.	
2. El Ingeniero de Requerimientos selecciona el requerimiento que desea eliminar y presiona el botón "Delete".	**3.** El sistema muestra un mensaje para confirmar la eliminación del requerimiento.
4.El Ingeniero de Requerimientos presiona el botón "OK".	**5.** El sistema registra la eliminación del requerimiento.
6. El Ingeniero de Requerimientos presiona el botón "Continue".	**7.**El sistema muestra la página correspondiente a Evaluar Requerimientos

Tabla 5: Descripción caso de uso Evaluar Requerimiento.

Acción del actor	**Respuesta del sistema**
1. Este caso de uso comienza cuando el Ingeniero de Requerimientos desea evaluar la calidad de los requerimientos.	
2. El Ingeniero de Requerimientos presiona el botón "Evaluate".	**3.** El sistema ejecuta el servicio de evaluación de calidad de los requerimientos previamente seleccionados y muestra los resultados.
4.El Ingeniero de Requerimientos presiona el botón "Print".	**5.** El sistema provee los mecanismos para la impresión del reporte con los resultados obtenidos.
6. El Ingeniero de Requerimientos presiona el botón "Verify".	**7.** El sistema muestra la página correspondiente a Verificar Requerimientos.

Tabla 6: Descripción caso de uso Verificar Requerimientos.

Acción del actor	**Respuesta del sistema**
1. Este caso de uso comienza cuando el Ingeniero de Requerimientos desea verificar los resultados de la evaluación.	
2. El Ingeniero de Requerimientos selecciona los requerimientos que desea verificar.	**3.**El sistema agrupa y prepara los requerimientos seleccionados para la verificación.
4.El Ingeniero de Requerimientos presiona el botón "Verify".	**5.** El sistema muestra los requerimientos seleccionados.
6. El Ingeniero de Requerimientos verifica los resultados en el reporte provisto por el sistema.	
7.El Ingeniero de Requerimientos presiona el botón "Print".	**8.**El sistema provee los mecanismos para imprimir el reporte resultante de la verificación.
Curso Alternativo	
Acción del actor	
6.a El Ingeniero de Requerimientos presiona el botón "Editar". El sistema registra los resultados correctamente.	
6.b El Ingeniero de Requerimientos presiona el botón "Eliminar". El sistema registra la eliminación correctamente.	

IV

Especificación de Requerimientos de Software

A continuación se detallan los requerimientos de software contemplados para el sistema de servicios bancarios propuestos por Saavedra (2016).

.1/ Home Banking System

.1.1/ Introduction

This Software Requirements Specification (SRS) specifies the requirements of the Home Banking System (HBS), which will be used in the Banks. This document will be utilized by the client to make sure all specifications are correct and verified by the software engineer to design the system.

.1.2/ Purpose

This Software Requirements Specification (SRS) specifies the requirements of the Home Banking System (HBS), which will be used in the Banks and by its clients. This document will be utilized by the client to make sure all specifications are correct and verified by the software engineer to design the system.

.1.3/ Scope

As competition has intensified and client needs have also increased, so have the challenges faced by banks. The pressure on margins has increased, often driven by new market entrants with lower cost bases, and the avalanche of new regulations and compliance seems to gather pace daily. Clients demand access to their financial information regardless of their location or the time of day, and if their current financial institution can't provide it they can always go to someone else who can.

Decades ago, legacy core banking systems were often installed, but these systems just cannot cope with supporting the latest products and, when it is possible, the process is complex, time-consuming, and expensive. Just keeping these systems running can often consume more than 70% of the IT budget leaving little money to gain advantage over competitors. Gaining insight into the client's needs can be extremely difficult, involving the collation of a large amount of data from disparate systems held in different formats. And by the time the data is collected, it is often too late – the clients' needs have moved on.

For the smooth working of the bank, a solution needs to be designed in such a way that, all the operations that were previously performed with difficulties are performed easily in this system. For the clients, an internet solution is the most appropriate one as almost all clients have access to it.

The Home Banking System deals with clients. It facilitates the clients to perform a wide range of functions like fund transfer, checkbooks request, viewing account statement, etc. over the internet. This reduces the inconvenience of having to go to the bank for each and everything. The client is no longer considered 'Client of the Branch' but the 'Client of the No Geographical Limits Bank'.

.1.4/ Product Overview

.1.4.1/ Product perspective

Our Product consists of a Home Banking System (HBS). The HBS must be exclusively for the clients, who must be access it from anywhere having an internet connection. The HBS uses well designed Web Forms for specific actions required by the users. This system connects to a main database server for storing and retrieving the data of the clients.

.1.4.2/ Product functions

Main functions of the HBS System:

1. Login System of HBS.
2. Change User Password.
3. Viewing the Accounts Linked to a username.
4. Viewing the Transaction Summary of an Account.
5. Fund Transfer among Accounts.
6. Checkbooks and Deposit Tickets Ordering System.
7. Add to and Consult a bank statement.
8. Make a payment.
9. Consult purchases in shops.
10. Establish a time deposit.
11. Request a pre-approved loan.
12. Consult details of credit cards.
13. Request subscription to an insurance against theft at an ATM.
14. Consult consolidated position.
15. Configure messages service and alerts.
16. Modify personal data.
17. Logout System (HBS).

.1.4.3/ User characteristics

BANK CLIENTS: The clients use the HBS.

.1.4.4/ Limitations

Design and implementation constraints:

1. Enhancements to the security features might lead to a performance overhead.
2. Recommended bandwidth is 64 KBps.
3. Central Server must be online round the clock.

.1.4.5/ References

Language Extended Lexicon (LEL)

.1.4.6/ Specific requirements

TBD

.1.4.7/ External interfaces

TBD

.1.4.8/ System interfaces

TBD

.1.4.9/ User interfaces

The first page of HBS must be a login screen which asks for the customer's username and password and, on successful verification, it transfers the user to the main page where the user can see all his/her linked accounts. The user must be able to use the different functionalities of the HBS by clicking on various links that are provided on each page. There must be a logout link, that when pressed, the user is logged out of the HBS immediately.

.1.4.10/ Hardware interfaces

TBD

.1.4.11/ Software interfaces

The product is implemented in the Linux Operating System Environment (Fedora Core 4).

The interface of the HBS must be designed using PHP5.

.1.4.12/ Communications interfaces

This product uses an internet connection to connect to the main database server of the bank.

.1.4.13/ Functions

.1.4.14/ Client User

1. The system shall allow the client to log in by entering username and password. (M) (H)
2. The system could allow changing the password.
3. The system shall allow the client to log out of the system.
4. The system shall allow the client to consult the UBK.
5. The system shall allow the client to print the UBK.
6. The system shall allow the client to download a file with the UBK.
7. The system shall allow the client to download a file with the Unique Bank Key.
8. The system shall allow the client to consult a balance.
9. The system shall allow a balance to be printed by the client.
10. The system shall allow the client to download a file with a balance.
11. The system shall allow the client to modify the daily withdrawal limit at ATMs for his/her debit card.
12. The system shall allow the client to print a modification voucher of the daily withdrawal limit at ATMs for his/her debit card.
13. The system shall allow the client to download a file with a modification voucher of the daily withdrawal limit at ATM for his/her debit card.
14. The system shall allow the client to request checkbooks, only if he holds a current account, by entering his/her account number, amount of checkbooks, and amount of checks per checkbook and checks type.
15. The system shan't allow the client to request checkbooks if he/she does not hold any current account (see functional requirement number 3.2.1.150).
16. The system shall allow the client to print a voucher of a checkbooks request.
17. The system shall allow the client to download a file with a voucher of a checkbooks request thus allowing to save and/or to print that voucher.
18. The system shall allow the client to request deposit tickets by selecting an account in a combo box.
19. The system shall allow the client to print a voucher of a deposit tickets request.
20. The system shall allow the client to download a file with a voucher of a deposit tickets request.
21. The system shall allow the client to add an account to the bank statement service.

22. The system shall allow the client to print a voucher of an adding of an account to the bank statement service.

23. The system shall allow the client to download a file with the accounts added to the bank statement service.

24. The system shall allow the client to remove an account added to the bank statement service.

25. The system shall allow the client to consult the latest bank statements.

26. The system shall allow the client to print the latest bank statements.

27. The system shall allow the client to download a file with the latest bank statements.

28. The system shall allow the client to consult a bank statement by entering the extract number notified via email by the bank.

29. The system shall allow the client to download a file with details of a bank statement.

30. The system shall allow the client to print details of a bank statement.

31. The system shall allow the client to consult his/her own accounts added to transfer.

32. The system shall allow the client to print details of his/her own accounts added to transfer.

33. The system shall allow the client to download a file with details of his/her own accounts added to transfer.

34. The system shall allow the client to consult third-party accounts added to transfer.

35. The system shall allow the client to print details of third- party accounts added to transfer.

36. The system shall allow the client to download a file with details of third-party accounts added to transfer.

37. The system shall allow the client to remove his/her own accounts added to transfer.

38. The system shall allow the client to print a voucher of a removal of his/her own accounts added to transfer.

39. The system shall allow the client to remove third-party accounts added to transfer.

40. The system shall allow the client to print a voucher of a removal of third-party accounts added to transfer.

41. The system shall allow the client to transfer funds from one source account to a target account, owned by him/her or third party, in this or any other bank.

42. The system shall allow the client to print a voucher of funds transferred between accounts.

43. The system shall allow the client to download a file with a voucher of funds transferred between accounts.

44. The system shall allow the client to consult a summary of funds transfers between accounts on a certain date range.

45. The system shall allow the client to print a summary of funds transfers between accounts on a certain date range.

46. The system shall allow the client to download a file with a summary of funds transfers between accounts on a certain date range.

47. The system shall allow the client to consult account movements on a certain date range.

48. The system shall allow the client to print account movements on a certain date range.

49. The system shall allow the client to download a file with account movements on a certain date range.

50. The system shall allow the client to consult the latest account movements.

51. The system shall allow the client to print the latest account movements.

52. The system shall allow the client to download a file with the latest account movements.

53. The system shall allow the client to make a payment either by entering the tax/service category, payment code and account or selecting tax or service added to the payments agenda and account.

54. The system shall allow the client to print a payment voucher.

55. The system shall allow the client to download a file with a payment voucher.

56. The system shall allow the client to consult payments concepts added to the payments agenda.

57. The system shall allow the client to print details of services or taxes added to the payments agenda.

58. The system shall allow the client to download a file with details of services or taxes added to the payments agenda.

59. The system shall allow the client to add a service or tax to the payments agenda.

60. The system shall allow the client to print a voucher of a service or tax added to the payments agenda.

61. The system shall allow the client to remove a concept added to the payments agenda.

62. The system shall allow the client to print a voucher of the removal of a service or tax added to the payments agenda.

63. The system shall allow the client to consult payments made using an account.

64. The system shall allow the client to print details of payments made using an account.

65. The system shall allow the client to download a file with details of payments made using an account.

66. The system shall allow the client to visualize a voucher of payments made using an account.

67. The system shall allow the client to reprint a voucher of payments made using an account.

68. The system shall allow the client to download a file with a voucher of payments made using an account.

69. The system shall allow the client to consult the latest purchases in shops.

70. The system shall allow the client to print the latest purchases in shops.

71. The system shall allow the client to download a file with the latest purchases in shops.

72. The system shall allow the client to consult the purchases made on a certain date range.

73. The system shall allow the client to print purchases made on a certain date range.

74. The system shall allow the client to download a file with purchases made on a certain date range.

75. The system shall allow the client to consult holdings of time deposits.

76. The system shall allow the client to print holdings of time deposits.

77. The system shall allow the client to download a file with holdings of time deposits.

78. The system shall allow the client to consult investment rates of a time deposit.

79. The system shall allow the client to print investment rates of a time deposit.

80. The system shall allow the client to download a file with investment rates of a time deposit.

81. The system shall allow the client to establish a time deposit.

82. The system shall allow the client to print a voucher of establishment of a time deposit.

83. The system shall allow the client to download a voucher of establishment of a time deposit.

84. The system shall allow the client to request an automatic renewal of a time deposit.

85. The system shall allow the client to print a request for automatic renewal of a time deposit.

86. The system shall allow the client to perform an early cancellation of a time deposit.

87. The system shall allow the client to print a voucher of early cancellation of a time deposit.

88. The system shall allow the client to change the accreditation account of a time deposit.

89. The system shall allow the client to print a voucher of a change of accreditation account of a time deposit.

90. The system shall allow the client to request a pre-approved loan.

91. The system shall allow the client to print a voucher of a pre-approved loan request.

92. The system shall allow the client to download a voucher of a pre-approved loan request.

93. The system shall allow the client to consult acquired loans.

94. The system shall allow the client to print details of acquired loans.

95. The system shall allow the client to download a file with details of acquired loans. 3.2.1.96 The system shall allow the client to consult credit cards details.

96. The system shall allow the client to print credit cards details.

97. The system shall allow the client to download a file with credit cards details.

98. The system shall allow the client to consult the credit limit and available credit on a credit card.

99. The system shall allow the client to consult the closing date and payment due date of a credit card.

100. The system shall allow the client to consult the monthly credit card consumption.

101. The system shall allow the client to print the monthly credit card consumption.

102. The system shall allow the client to download a file with the monthly credit card consumption.

103. The system shall allow the client to consult the latest credit card summary.

104. The system shall allow the client to print the latest credit card summary.

105. The system shall allow the client to download a file with the latest credit card summary.

106. If the client is the holder of a credit card, then the system shall allow the client to request an additional credit card typically by entering: name, surname, document number, date of birth, etc. of the additional holder.

107. The system shall allow the client to print a voucher of an additional credit card request.

108. The system shall allow the client to consult additional credit card requests.

109. The system shall allow the client to print additional credit card requests.

110. The system shall allow the client to download a file with additional credit card requests.

111. The system shall allow the client to request cash advances using a credit card.

112. The system shall allow the client to print a voucher of cash advances request using a credit card.

113. The system shall allow the client to download a file with the voucher of cash advances requests using a credit card.

114. The system shall allow the client to request a subscription to an insurance against theft at an ATM.

115. The system shall allow the client to print a voucher of a subscription to an insurance against theft at an ATM.

116. The system shall allow the client to consult his/her position in the bank at December 31 for the selected year.

117. The system shall allow the client to print his/her position in the bank at December 31 for the selected year.

118. The system shall allow the client to download a file with his/her position in the bank at December 31 for the selected year.

119. The system shall allow the client to consult his/her consolidated position, in my opinion.

120. The system shall allow the client to print his/her consolidated position.

121. The system shall allow the client to download a file with his/her consolidated position.

122. The system shall allow the client to configure messages service and alerts.

123. The system shall allow the client to modify his/her personal data.

124. The system shall allow the client to modify his/her personal data.

125. Usability Requirements

126. The system must be easy to use.

.1.5/ Performance requirements

The most important factor in the working of the whole project is its Connectivity with the Server and the Mode of Connection. If the connection is a 64 Kbps one then the performance of the Products (HBS) will be much better than one with a 10 Kbps connection.

.1.6/ Logical database requirements

TBD

.1.7/ Design constraints

TBD

.1.8/ Software system attributes

.1.9/ Availability

127. The system must be accessible from any geographic location 24 hours a day.

.1.9.1/ Security

1. The system shall allow the client to change password at the first login.
2. The system shall control the amount of incorrect accesses, blocking the user account in case of exceeding 3 failed attempts.
3. The system shall allow using a virtual keyboard to enter the password (Associated with RF 3.2.1.1).
4. The system must have a website that uses a secure channel.
5. The data transmitted by the website must be encrypted.
6. The system must produce an auto-disconnect when the client PC remains inactive for a time.

.1.10/ Reliability

TBD

.1.11/ Maintainability

TBD

.1.12/ Portability

TBD

.1.12.1/ Supporting information

TBD

.1.13/ Verification

TBD

.1.14/ Appendices

TBD

.1.14.1/ Assumptions and dependencies

We have made the following assumptions:

- The main server must never go offline.

SPIM
école doctorale **sciences pour l'ingénieur et microtechniques**

Titulo: Framework para la Evaluación Automática de la Calidad de los Requerimientos de Software basado en Redes Neuronales

Palabras clave: Deep learning, Quality, Software Requirements, Neural Networks

Resumen:

El lenguaje natural es ampliamente utilizado para obtener y documentar los requerimientos de software. Aunque se complementa con modelos y notaciones formales, su uso prevalece en la mayoría de las industrias debido a su flexibilidad y comprensibilidad. Sin embargo, su naturaleza flexible y expresiva puede conducir a inconsistencias, redundancias y ambigüedades en los requerimientos, lo cual afecta negativamente al ciclo de vida del software, dado que puede conducir a la concepción de errores de baja calidad. Para abordar este problema, es necesario implementar mecanismos de control de calidad desde las etapas iniciales del proyecto. La calidad de los requerimientos es considerada un factor determinante de éxito para un proyecto de software. En este sentido, la integración de la Inteligencia Artificial (IA) ha mostrado resultados prometedores en la Ingeniería de Requerimientos, optimizando tareas intensivas. La IA ha experimentado un crecimiento notable debido a su aplicación en diversos dominios, respaldada por grandes conjuntos de datos, entrenamiento escalable y redes neuronales avanzadas. La combinación de técnicas tradicionales y de IA en la Ingeniería de Requerimientos ha sido analizada en un mapeo sistemático, revelando oportunidades de investigación en el aprendizaje supervisado aplicado a esta disciplina. En este sentido, se desarrolló un conjunto de modelos neuronales que permiten evaluar automáticamente la calidad de los requerimientos, utilizando un corpus de requerimientos de software que abarca diferentes dominios. Los modelos neuronales se enfocan en evaluar la calidad de los requerimientos expresados en lenguaje natural, siguiendo los estándares ISO/IEC/IEEE 29148:2018. Este enfoque ofrece mecanismos automáticos de control y verificación de la calidad de los requerimientos, beneficiando a ingenieros y profesionales del software para obtener requerimientos de calidad a fin de cumplir con las expectativas de los clientes.

UTN
UNIVERSIDAD TECNOLÓGICA NACIONAL

Universidad Tecnológica Nacional
Facultad Regional Santa Fe
Lavaisse 610
S3004, Santa Fe, Argentina