Hacia la Evaluación Automática de la Calidad de los Requerimientos de Software usando Redes Neuronales Long Short Term Memory

María Guadalupe Gramajo[1,2], Luciana Ballejos[1], and Mariel Ale[1]

[1] CIDISI-UTN-FRSF, Lavaisse 610, Santa Fe, Argentina
[2] GITIA-UTN-FRT, Rivadavia 1050, Tucumán, Argentina
{mgramajo,lballejo,male}@frsf.utn.edu.ar

Abstract. El primer paso para obtener un producto de software de calidad es abordar la calidad de los requerimientos. Dado que los requerimientos están escritos en lenguaje natural, la flexibilidad y las características propias del lenguaje pueden conducir a errores durante su especificación y, en consecuencia, esto puede influir negativamente en fases posteriores del ciclo de vida del software. Por esta razón, en este artículo se propone un enfoque innovador que combina técnicas de procesamiento del lenguaje natural y redes neuronales, específicamente redes neuronales recurrentes LSTM, para predecir la calidad textual de los requerimientos. Inicialmente, se aborda el análisis de la propiedad de calidad singular definida en el estándar ISO/IEC/IEEE 29148: 2018. El modelo neuronal propuesto es entrenado con un conjunto de datos que incluye 1000 requerimientos. Cada requerimiento es sometido a un proceso de tokenización y etiquetado gramatical, con el fin de obtener una secuencia representativa que actúe de entrada al modelo neuronal. La propuesta proporciona resultados prometedores con un alto nivel de precisión, lo que motiva explorar su aplicación en la evaluación de otras propiedades de calidad.

Keywords: Requirements Engineering · Deep Learning · Long Short Term Memory.

1 Introducción

En los últimos años, la Inteligencia Artificial (IA) se ha posicionado como una herramienta poderosa y accesible, capaz de ser utilizada como un componente clave en el desarrollo de sistemas de software. Esto es así, debido al surgimiento de nuevas estrategias de aprendizaje, disponibilidad de mayores conjuntos de datos y el aumento de la capacidad de procesamiento en los ordenadores [1].

La integración de las tecnologías de IA en la Ingeniería del Software tiene por objetivo optimizar el proceso de desarrollo de productos de software y automatizar tareas que demandan esfuerzos intensivos, a fin de obtener sistemas con alta calidad, uno de los propósitos más pretendidos en el área mencionada. En relación a esto, desde hace tiempo se ha reconocido que el éxito de un proyecto

de desarrollo de software depende en gran medida de la calidad de los requerimientos [1–4]. Consecuentemente, abordar la calidad de los requerimientos es el primer paso para obtener un producto de software de calidad.

A menudo, diversos problemas relacionados con la calidad textual de los requerimientos surgen durante las tareas de obtención y especificación, dado que éstos son escritos en lenguaje natural. La flexibilidad y la naturaleza inherente del lenguaje hacen que los requerimientos estén propensos a inconsistencias, redundancias y ambigüedades y, consecuentemente, esto influye de manera negativa en las fases posteriores del ciclo de vida del software.

Dada la importancia del aseguramiento de la calidad de los requerimientos, esta temática ha sido estudiada desde diversas perspectivas. En la literatura se distinguen tres grandes categorías de investigación [5]. La primera categoría incluye aquellas herramientas desarrolladas para asistir al ingeniero de requerimientos respecto a la escritura, señalando debilidades y proporcionando indicaciones acerca de cómo los requerimientos pueden ser mejorados. Ejemplos de estas herramientas son RQA (Requirements Quality Analyzer) [6], ARM (Automated Requirement Measurement) [7], QuARS (Quality Analyzer for Requirements Specifications) [8] y RQV (Requirement Quality Verification Tool) [9]. La segunda categoría incluye propuestas enfocadas en evaluar la calidad transformando los requerimientos textuales en modelos formales mediante el análisis lingüístico y el uso de reglas gramaticales [10–13]. La tercera categoría distingue las propuestas que utilizan razonamiento basado en reglas y técnicas de aprendizaje automático para clasificar los requerimientos, y consecuentemente, determinar si éstos son de buena o mala calidad [14–18]. Cabe señalar que algunas propuestas cubren el análisis y evaluación de un conjunto completo de atributos de calidad, tomando como referencia modelos y estándares, mientras que otras abordan el estudio particular de ciertas propiedades individuales de calidad.

Inicialmente, en este artículo se aborda el análisis de la propiedad de calidad singular definida en el estándar ISO/IEC/IEEE 29148:2018. Dicho estándar establece que un requerimiento cumple con esta propiedad si incluye en su especificación la definición de una única funcionalidad, característica, restricción o factor de calidad. Aunque a pesar de ello, pueden existir múltiples condiciones bajo las cuales se debe cumplir el requerimiento [19]. La evaluación y el análisis de esta propiedad es esencial, ya que su obtención está asociada a otras características de calidad deseables, tales como la precisión, abstracción, trazabilidad y la propiedad modificable. Por lo tanto, alcanzar la singularidad de los requerimientos influye positivamente en la consecución de otras propiedades de calidad [20].

En la literatura, son pocas las propuestas que abordan el análisis de la propiedad de calidad singular [9],[15],[17],[18],[20]. La mayoría de los artículos coinciden en el uso de técnicas de procesamiento de lenguaje natural y mecanismos de control, tales como reglas gramaticales, expresiones regulares, corpus de palabras y/o frases para obtener secuencias representativas de los requerimientos y luego, procesarlos mediante la ejecución de reglas y sentencias condicionales definidas por un experto. El uso de instrucciones secuenciales y condicionales

conduce a procesos de evaluación estáticos que no poseen la capacidad de considerar excepciones gramaticales y/o características propias de un determinado dominio.

En contraste a las propuestas mencionadas, en este artículo se presenta un enfoque más flexible, que utiliza redes neuronales recurrentes (RNNs) para predecir la calidad de los requerimientos. Para ello, las RNNs generan internamente reglas dinámicas que asocian la estructura gramatical y sintáctica de los requerimientos con el criterio de clasificación de un grupo de expertos, respecto a la propiedad de calidad bajo análisis. Estas reglas se obtienen luego de entrenar la red neuronal con un conjunto de requerimientos previamente clasificados por los expertos. Específicamente, se utilizan redes neuronales de memoria de largo y corto plazo (LSTM), un tipo de RNNs, porque son apropiadas para procesar información secuencial. Esta característica de las LSTM, permite capturar la naturaleza secuencial inherente de las expresiones en lenguaje natural utilizadas para especificar los requerimientos de software [21]. En resumen, las redes neuronales LSTM permiten el análisis de los requerimientos expresados en lenguaje natural, considerando la dependencia secuencial de los elementos que los componen. Y, a partir de la relación entre la secuencialidad gramatical y los elementos que componen a los requerimientos, la red neuronal genera conocimiento que luego, utilizará para producir resultados.

El resto del contenido de este artículo se organiza de la siguiente manera. En la sección II se describe la metodología aplicada. En la sección III se presenta un caso de estudio. La sección IV presenta discusiones sobre los resultados obtenidos. Por último, en la sección V se detallan las conclusiones y trabajo futuro.

2 Metodología

Esta sección describe el conjunto de datos utilizado y el método aplicado para realizar el preprocesamiento de datos. La arquitectura de la red neuronal también es descripta en esta sección.

2.1 Descripción del Conjunto de Datos

Como se mencionó en la sección anterior, el objetivo de esta propuesta consiste en predecir la calidad de los requerimientos de software expresados en lenguaje natural. Para ello, se definió un modelo neuronal que adopta un enfoque de aprendizaje del tipo supervisado. A fin de proveer los datos necesarios para su entrenamiento, se conformó un corpus compuesto por 1000 requerimientos extraídos del conjunto de datos *Public Requirements Dataset* (PURE) [22]. Este conjunto de datos proporciona 79 documentos de requerimientos expresados en lenguaje natural, los cuales fueron extraídos de la Web. Los documentos contenidos en PURE están escritos en inglés y cubren múltiples dominios procedentes de diversas compañías y proyectos universitarios, lo que contribuye a la heterogeneidad de los datos incluidos en el corpus.

Los requerimientos que conforman el corpus fueron seleccionados de manera aleatoria y extraídos manualmente desde 26 documentos. Sólo se incluyeron requerimientos expresados en texto plano y se excluyeron aquellos definidos como enlaces a documentos externos. 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 el modelo neuronal.

Posteriormente, los requerimientos fueron sometidos a un proceso de clasificación manual para ser identificados como singulares o no singulares en base al juicio de un grupo de expertos. El proceso de clasificación fue realizado por 3 investigadores en el área de la Ingeniería de Requerimientos. Los requerimientos fueron clasificados utilizando la etiqueta "1" y "0" para representar la "singularidad" o "no singularidad" respectivamente. A partir de ello, es posible establecer el conjunto de datos que se utilizará durante el entrenamiento del modelo.

2.2 Preprocesamiento de Datos

Los requerimientos contenidos en el corpus fueron sometidos a un proceso de tokenización y POS (Part Of Speech) tagging. Este proceso permite estructurar los requerimientos en un formato de entrada adecuado para el modelo neuronal propuesto. Los requerimientos fueron tokenizados, en primer lugar, a nivel de oración y, posteriormente, a nivel de palabras, con la librería Natural Language Toolkit (NLTK) [23]. Esta librería permite segmentar texto en oraciones y palabras.

A continuación, se realizó el proceso de etiquetado POS tagging, también conocido como etiquetado gramatical. Este proceso consiste en asociar los tokens que conforman una oración con su etiqueta gramatical correspondiente. El conjunto de etiquetas utilizado es el proporcionado por Stanford [24].

Luego, las etiquetas son listadas dinámicamente en un rango finito de enteros según su ocurrencia en las representaciones de requerimientos a nivel de etiquetas. Con esto último, se hace referencia a que cada palabra etiquetada en una oración tiene correspondencia con un número entero. Con ello, las representaciones de los requerimientos a nivel de etiquetas pueden ser transformadas en secuencias de enteros.

Otro aspecto a destacar, es que los requerimientos que componen el corpus no poseen la misma longitud. Por ello, es necesario trabajar sobre este aspecto para construir un conjunto de entradas homogéneo, en el que todos los requerimientos posean igual longitud. En esta propuesta se ha definido una longitud máxima de 100 palabras por requerimiento, considerando la mayor longitud presente en el corpus de requerimientos definido anteriormente. En aquellos casos donde los requerimientos tienen una longitud menor a la establecida, se adoptó la estrategia de añadir ceros a la izquierda a fin de completar la secuencia de enteros. Esta estrategia se denomina *pre sequence padding* y es ampliamente utilizada en problemas de predicción con datos de entrada de longitud variable.

Las representaciones numéricas obtenidas son almacenadas en forma de matriz con sus índices correspondientes. Éstas actúan de entrada al modelo neuronal propuesto para llevar a cabo el proceso de entrenamiento.

A continuación, la Figura 1 ilustra el proceso descripto a través de la representación de un requerimiento singular.

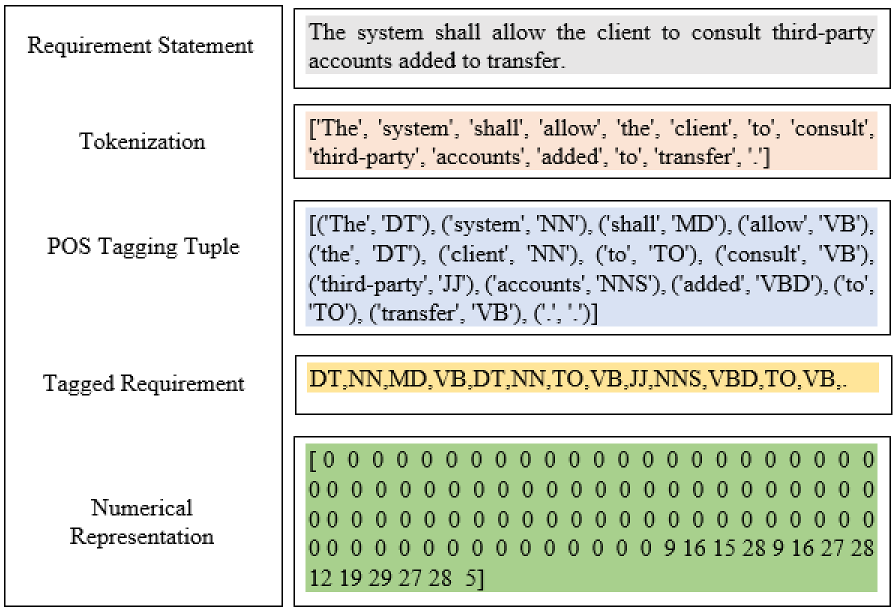


Fig. 1: Preprocesamiento de datos de Requerimiento Singular.

La Figura 1 muestra el preprocesamiento de datos de un requerimiento singular. En primer lugar, se observa la definición del requerimiento contenido en el corpus. Luego, el requerimiento es sometido a un proceso de tokenización, lo cual permite analizar individualmente cada palabra que conforma la oración. A continuación, se realiza el etiquetado gramatical o POS tagging, es decir, cada palabra es asociada a su etiqueta gramatical correspondiente. Con ello, se obtiene la representación del requerimiento a nivel de etiquetas. Finalmente, para lograr la representación numérica del requerimiento, las etiquetas fueron listadas dinámicamente. Es por ello, que su numeración está relacionada a la ocurrencia de todas las etiquetas presentes en el corpus. Cabe señalar, que el preprocesamiento de datos se realiza sobre el conjunto de requerimientos de forma automática.

La Figura 2 ilustra el preprocesamiento de datos de un requerimiento no singular. En dicha figura, se muestra un requerimiento clasificado de acuerdo al criterio de los expertos como no singular, dado que presenta cláusulas abiertas en su definición. La presencia de cláusulas abiertas, según lo establecido por INCOSE, conduce a problemas de interpretación y a la definición de requerimientos no verificables, que deben evitarse a fin de alcanzar la propiedad singular [25].

A pesar de no cumplir con la propiedad de calidad singular, se destaca que el preprocesamiento de datos se realiza de la misma manera sobre todo el corpus de requerimientos.

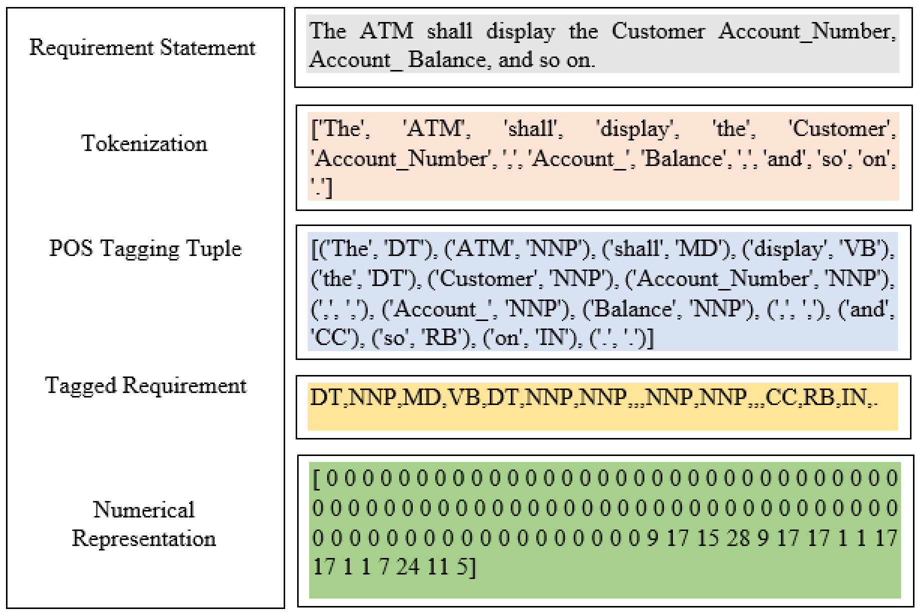


Fig. 2: Preprocesamiento de datos de Requerimiento No Singular.

2.3 Arquitectura de la Red Neuronal

En esta subsección se describe la arquitectura de la red neuronal propuesta y la configuración inicial de los parámetros incluidos.

La arquitectura de la red neuronal que se presenta incluye 38 *unidades de entrada* (*inputs*). Estas unidades permiten el ingreso de los requerimientos, previamente preprocesados y contenidos en un formato de entrada adecuado, a la *capa embebida* (*embedding layer*). Dicha capa se compone de dos elementos, el primero es la *dimensión de entrada* (*input dimension*) que representa el tamaño del vocabulario de los datos. Particularmente, el tamaño del vocabulario definido en esta propuesta tiene un valor igual a 38, lo cual tiene correspondencia con las etiquetas utilizadas para representar los requerimientos. El segundo elemento es la *dimensión de salida* (*output dimension*), la cual corresponde al tamaño del espacio vectorial en el que los requerimientos serán embebidos. Específicamente para este enfoque, la salida tiene un valor igual a 128.

La arquitectura presentada también incluye una capa LSTM, la cual permite el procesamiento de eventos secuenciales en la red neuronal. Esta capa

LSTM se encuentra posteriormente enlazada a una *capa totalmente conectada* (*full connected layer*). Cabe destacar que esta última, es la encargada de otorgar la salida del modelo. Ambas capas mencionadas están compuestas por tres parámetros: *número de unidades de entrada* (*num-input*), *número de unidades de salida* (*num-output*) y *número de capas* (*num-layers*). Los parámetros de configuración son los siguientes:

1. La capa LSTM posee 128 unidades de entrada, dado que considera el valor de la dimensión de salida de la capa embebida. El número de unidades de salida de la capa LSTM tiene un valor igual a 128. Mientras que, la cantidad de capas tiene un valor 1.

2. La *capa totalmente conectada* (*full connected layer*), posee 128 unidades de entrada correspondiente a las unidades de salida de la capa LSTM. Mientras que, el número de unidades de salida tiene valor 2, lo que permite representar las clases definidas en esta propuesta (singular - no singular). La cantidad de capas tiene valor 1. Además, las unidades de esta capa están sujetas a una *probabilidad de caída (dropout)* de 0.3. Por último, la *función softmax* determinar la probabilidad de acierto de cada clase definida.

A continuación, la Figura 3 representa la arquitectura de la red neuronal descripta y la Tabla 1 resume la configuración inicial de los parámetros.

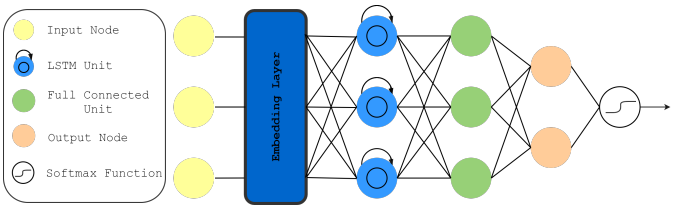


Fig. 3: Arquitectura Red Neuronal.

Tabla 1: Configuración Inicial de Parámetros.

Parámetro	Valor
Input Unit	38
Input Dimension Embeding Layer	38
Output Dimension Embeding Layer	128
Layer Type	LSTM
Input LSTM Layer	128
Output LSTM Layer	128
Input Dimension Full Connected Layer	128
Output Dimension Full Connected Layer	2
Dropout	0.3

Posteriormente, los parámetros fueron sometidos a un proceso de ajuste, a través del método de búsqueda *random grid search* para encontrar la combinación de valores paramétricos que devuelve el mejor rendimiento. Esta técnica consiste en establecer una cuadrícula de valores paramétricos y seleccionar combinaciones aleatorias para entrenar el modelo, a fin de analizar el rendimiento obtenido. El número de iteraciones de búsqueda se establece en función de tiempo/recursos.

3 Caso de Estudio

En esta seccción, se introduce un caso de estudio con el objetivo de ilustrar el enfoque propuesto.

A continuación, se presenta la descripción de dos requerimientos incluidos en el corpus con los cuales, se ilustra el flujo de trabajo del enfoque presentado en este artículo. En primer lugar, se observa la declaración de un requerimiento no singular, clasificado como tal de acuerdo al criterio de los expertos. Esto es así, dado que en su declaración se emplean pronombres que pueden conducir a ambigüedades y que no contribuyen a la obtención de la propiedad singular. Posteriormente, se realiza el proceso de tokenización del requerimiento con lo que se obtiene el requerimiento segmentado en oraciones y palabras. Luego, se realiza el etiquetado gramatical del requerimiento. La Figura 4 ilustra el proceso de tokenización y POS tagging obtenido.

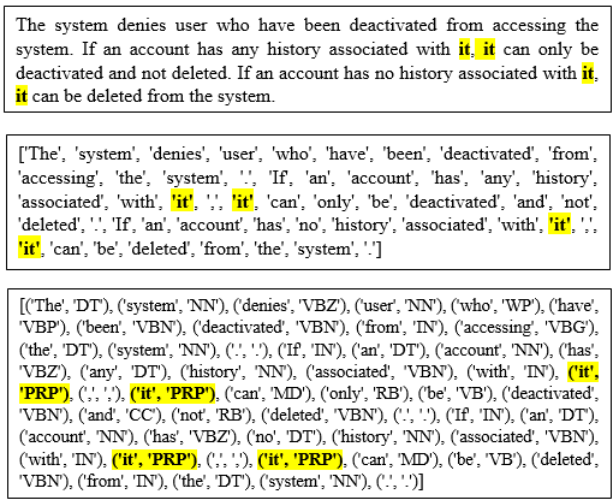


Fig. 4: Análisis de Requerimiento No Singular.

En la Figura 5 se observa la declaración de un segundo requerimiento. La clasificación otorgada por los expertos corresponde a un requerimiento singular.

A pesar de observarse la presencia de pronombres, los cuales pueden conducir a ambigüedades, éste es clasificado como un requerimiento singular. Esto es así, dado que los expertos consideran que para este caso particular, la existencia de pronombres en la declaración del requerimiento no genera inconvenientes en su comprensión, y por lo tanto, no es relevante. Este hecho, demuestra que la clasificación no es un proceso estático, y que existen ciertas consideraciones y excepciones sobre las que un experto puede reflexionar al clasificar un requerimiento.

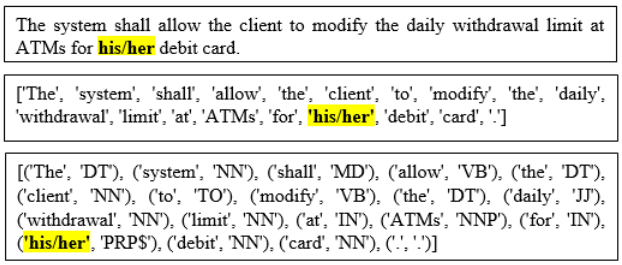


Fig. 5: Análisis de Requerimiento Singular.

Al extrapolar esta situación al corpus de requerimientos, se obtendrá un conjunto de excepciones y consideraciones sobre las que los expertos pueden reflexionar al realizar el proceso de clasificación. En este aspecto, las redes neuronales LSTM tienen un rol fundamental debido a su capacidad de procesamiento secuencial. Esta característica permite analizar la dependencia secuencial de los elementos (palabras/etiquetas/representaciones numéricas) que conforman las expresiones en lenguaje natural (requerimientos). A partir de ello, la red neuronal aprende a asociar la secuencialidad de las etiquetas gramaticales que representan a los requerimientos con la clasificación otorgada por los expertos (singular - no singular). Consecuentemente, la red neuronal genera conocimiento sobre las excepciones presentes en el corpus. Con lo que, la red neuronal toma en cuenta no sólo las relaciones entre la secuencialidad gramatical y el criterio de clasificación, sino que, además, considera la estructura gramatical y sintáctica de los requerimientos para realizar la predicción de calidad. Estas características permiten la flexibilidad del enfoque, en contraste, a otras propuestas presentes en la literatura.

Dado que los requerimientos deben ser estructurados en un formato de entrada adecuado para ser procesados por la red neuronal, éstos son transformados desde la representación a nivel de etiquetas a una representación numérica, tal como se menciona en la sección 2. La Figura 6 ilustra la representación numérica de ambos requerimientos descriptos. Posteriormente, las representaciones numéricas obtenidas se almacenan en forma de matriz con sus índices correspondientes. Éstas últimas, actúan de entrada a la red neuronal para su entrenamiento.

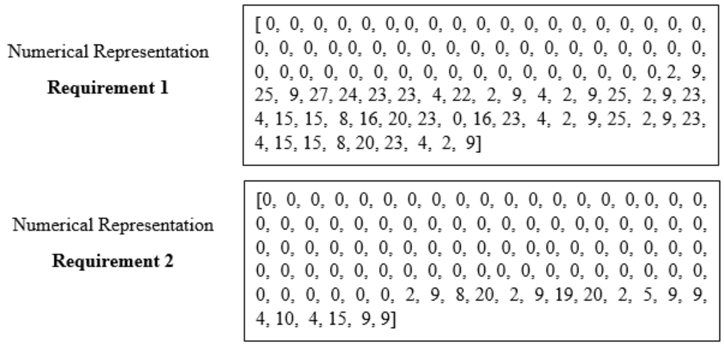


Fig. 6: Representación Numérica de Requerimientos.

Luego de realizar el preprocesamiento de los requerimientos, éstos son sometidos a *técnicas de validación cruzada (k-fold cross validation)*, las cuales permiten preparar el conjunto de datos para entrenar la red neuronal múltiples veces y, así, evitar el *sobreajuste (overfitting)* del modelo. El número de *particiones (folds)* se ha establecido en k=10. Esto se refiere a que la muestra de datos se ha dividido en k partes iguales. Cada partición considera la relación (80-10-10), es decir, el 80% de los datos se utiliza para el entrenamiento del modelo. Mientras que, el 10% se utiliza para la prueba y el 10% restante para la validación. Respecto al número de iteraciones se ha establecido en i=10. Al finalizar una iteración, cada partición conmuta aleatoriamente su conjunto de entrenamiento, prueba y validación manteniendo la relación (80-10-10).

En esta propuesta se aplica, además, la técnica *random grid search*, a fin de optimizar los hiperparámetros. Esta técnica permite ejecutar el modelo neuronal con un conjunto de configuraciones aleatorias, lo que facilita posteriormente, identificar la configuración que retorna el mejor rendimiento. Los hiperparámetros considerados poseen un rango de valores previamente definidos, con los que la red neuronal es ejecutada, y de esta manera puede asumir múltiples combinaciones aleatorias posibles. Cabe señalar, que esta propuesta considera una segunda variante de RRNNs, denominada GRU, Gated Recurrent Unit como hiperparámetro en las múltiples combinaciones aleatorias [26]. La Tabla 2 resume los hiperparámetros considerados en el modelo neuronal.

A continuación, la mejor configuración obtenida luego de ejecutar la técnica *random grid search* se presenta en la Tabla 3.

La ejecución de la mejor configuración obtenida se muestra en la Figura 7. En esta figura se observa el número de epochs, la pérdida de entrenamiento, la precisión en la validación, pérdida en la validación y duración de la ejecución. Es posible observar que el modelo alcanza la mayor precisión en el epoch 26. En este último, la precisión alcanzada es del 79% luego de 1900 ejecuciones del modelo. En cuanto a la pérdida en la fase de validación, su valor óptimo es alcanzado en el epoch 29. A partir de los resultados obtenidos, es posible determinar

Tabla 2: Configuración de Hiperparámetros.

Hiperparámetro	Rango
Epoch	[3, 4, 5, 10, 30, 40]
Learning Rate	[0.1, 0.01, 0.001]
Embedding Dimension	[64, 128, 256]
Number Layers	[1, 2]
Number Units	[64, 128, 256]
Dropout	[0, 0.1, 0.3]
Layer Type	LSTM , GRU
Optimizer	Adam
Error Criteria	Binary Cross Entropy

Tabla 3: Mejor configuración.

Hiperparámetro	Valor
Epoch	30
Learning Rate	0.01
Embedding Dimension	128
Number Layers	2
Number Units	128
Dropout	0.3
Layer Type	LSTM

epoch	train_loss	valid_acc	valid_loss	dur
1	2.0426	0.5000	0.7397	0.3161
2	0.7639	0.5000	0.7065	0.0266
3	0.6790	0.5625	0.6867	0.0212
4	0.6841	0.5312	0.6892	0.0215
5	0.6872	0.5156	0.7002	0.0200
6	0.6940	0.5156	0.6970	0.0231
7	0.6963	0.5312	0.6993	0.0217
8	0.6966	0.5312	0.6893	0.0208
9	0.6907	0.5312	0.6813	0.0211
10	0.6794	0.5625	0.6673	0.0211
11	0.6753	0.6094	0.6495	0.0207
12	0.6715	0.6094	0.6505	0.0206
13	0.6632	0.6094	0.6595	0.0224
14	0.6624	0.5938	0.6564	0.0210
15	0.6615	0.5938	0.6488	0.0212
16	0.6576	0.6406	0.6397	0.0207
17	0.6563	0.6875	0.6276	0.0214
18	0.6564	0.7188	0.6141	0.0205
19	0.6486	0.7500	0.6097	0.0213
20	0.6443	0.7500	0.6028	0.0209
21	0.6303	0.7500	0.5796	0.0215
22	0.6217	0.7500	0.5730	0.0204
23	0.6050	0.7500	0.5588	0.0201
24	0.5911	0.7344	0.5454	0.0203
25	0.5707	0.7656	0.5398	0.0235
26	0.5569	0.7969	0.5245	0.0201
27	0.5422	0.7656	0.5130	0.0212
28	0.5287	0.7656	0.5062	0.0201
29	0.5151	0.7656	0.4932	0.0213
30	0.7479	0.6875	0.5726	0.0221

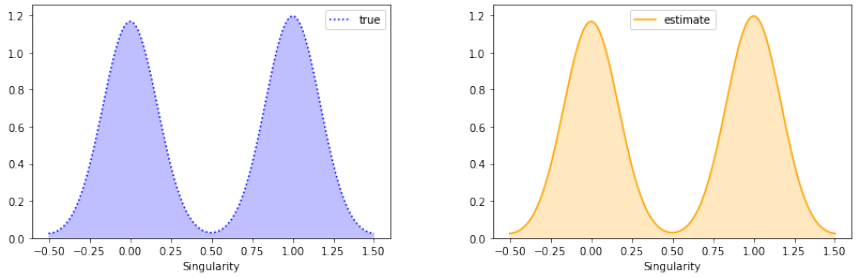
Fig. 7: Mejor Configuración Obtenida.

que esta configuración es prometedora, por lo tanto, motiva su exploración en la evaluación de otras propiedades de calidad, a fin de demostrar que el en-

foque propuesto puede dar soporte a la evaluación general de la calidad de los requerimientos documentados en una Especificación de Requerimientos.

Otro aspecto a mencionar son las características de hardware y software utilizadas en la ejecución del modelo neuronal. Estas son, Linux 4.14.137 SMP x86_64; RAM 12GB; GPU Google Compute Engine y Python 3.6.8.

Por último, se presentan dos gráficos para representar la distribución de los valores verdaderos del conjunto de datos y los valores predichos por el modelo. En primer lugar, la Figura 8a muestra la distribución de las clases de los requerimientos en el corpus, esto permite observar la proporción de las clases utilizadas para realizar el entrenamiento de la red neuronal. En segundo lugar, la Figura 8b muestra la distribución de los valores predichos por la red neuronal, en ella se observa que la proporción se mantiene constante respecto a los valores de verdad del primer gráfico, lo cual corresponde a una precisión alta y a una variación mínima.



(a) Distribución de Valores Verdaderos. (b) Distribución de Valores Predichos.

Fig. 8: Distribución Valores del Conjunto de Datos.

4 Discusiones

En esta sección se discuten los resultados obtenidos y las limitaciones del modelo neuronal propuesto. En primer lugar, cabe señalar que el alcance de esta propuesta se limita al análisis de la propiedad de calidad singular. Mientras que, el análisis de otras propiedades de calidad se propone como futura línea de investigación. Específicamente, se observa que el error de predicción varía en un rango promedio de - 0.05 a 0.05. La Figura 9 muestra la distribución del error.

Otra de las limitaciones que se pueden observar como resultado de la ejecución de este modelo neuronal es la presencia de falsos positivos y falsos negativos. Cuya existencia se asocia a la necesidad de enriquecer el conjunto de requerimientos utilizados durante el entrenamiento de la red neuronal.

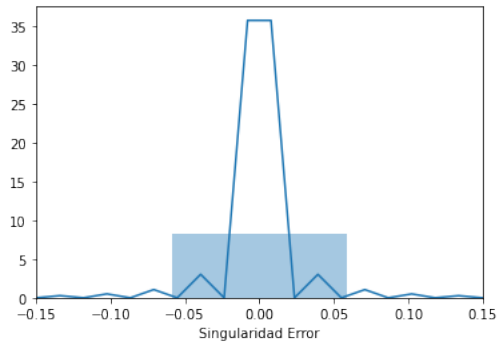


Fig. 9: Distribución del Error.

5 Conclusión

En este artículo se introdujo el uso de las redes neuronales recurrentes del tipo LSTM para abordar la evaluación de la propiedad de calidad singular de los requerimientos de software. Los requerimientos fueron preprocesados, a fin de estructurarlos en un formato de entrada adecuado para el modelo neuronal.

Esta propuesta utiliza un conjunto de datos compuesto por 1000 requerimientos para entrenar el modelo neuronal propuesto. El conjunto de datos generado ha sido particionado, considerando el 80% de la muestra para realizar el entrenamiento de la red neuronal. Mientras que, el 10% de la muestra se ha utilizado para la prueba y el 10% restante para la validación. La técnica de validación cruzada es aplicada en esta propuesta para evitar el sobreajuste del modelo y la técnica random grid search para optimizar los valores de los hiperpárametros.

El modelo neuronal alcanza su máximo rendimiento en el epoch 26. En este último, la precisión alcanzada es del 79%. En cuanto a la pérdida en la fase de validación su valor óptimo es alcanzado en el epoch 29. Los resultados obtenidos son prometedores e incentivan explorar su aplicación sobre otras propiedades de calidad.

Como trabajo futuro se pretende aplicar el enfoque propuesto a la evaluación y análisis del resto de las propiedades de calidad definidas en el estándar ISO/IEC/IEEE 29148-2018. Y luego, integrar la solución a un framework que permita la evaluación de la calidad de los requerimientos de manera automática.

6 Agradecimientos

Los autores agradecen el apoyo brindado por las siguientes instituciones: CONICET y Universidad Tecnológica Nacional (SIUTIFE0004923TC).

Referencias

1. R. Feldt, F. G. de Oliveira Neto, and R. Torkar, "Ways of applying artificial intelligence in software engineering," in *2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, pp. 35–41, IEEE, 2018.
2. B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Communications of the ACM*, vol. 31, no. 11, pp. 1268–1287, 1988.
3. M. I. Kamata and T. Tamai, "How does requirements quality relate to project success or failure?," in *15th IEEE International Requirements Engineering Conference (RE 2007)*, pp. 69–78, IEEE, 2007.
4. E. Knauss, C. El Boustani, and T. Flohr, "Investigating the impact of software requirements specification quality on project success," in *International Conference on Product-Focused Software Process Improvement*, pp. 28–42, Springer, 2009.
5. P. S. Kummler, L. Vernisse, and H. Fromm, "How good are my requirements?: A new perspective on the quality measurement of textual requirements," in *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pp. 156–159, IEEE, 2018.
6. W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, "Automated analysis of requirement specifications," in *Proceedings of the 19th international conference on Software engineering*, pp. 161–171, 1997.
7. A. Fantechi, S. Gnesi, G. Lami, and A. Maccari, "Applications of linguistic techniques for use case analysis," *Requirements Engineering*, vol. 8, no. 3, pp. 161–170, 2003.
8. F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool," in *Proceedings 26th Annual NASA Goddard Software Engineering Workshop*, pp. 97–105, IEEE, 2001.
9. F. Konig, L. C. Ballejos, and M. A. Ale, "A semi-automatic verification tool for software requirements specification documents," in *Simposio Argentino de Ingeniería de Software (ASSE)-JAIIO 46 (Córdoba, 2017).*, 2017.
10. M. Ilieva and O. Ormandjieva, "Automatic transition of natural language software requirements specification into formal presentation," in *International Conference on Application of Natural Language to Information Systems*, pp. 392–397, Springer, 2005.
11. K. Verma and A. Kass, "Requirements analysis tool: A tool for automatically analyzing software requirements documents," in *International semantic web conference*, pp. 751–763, Springer, 2008.
12. J. Holtmann, J. Meyer, and M. von Detten, "Automatic validation and correction of formalized, textual requirements," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, pp. 486–495, IEEE, 2011.
13. A. Arellano, E. Zontek-Carney, and M. A. Austin, "Frameworks for natural language processing of textual requirements," *International Journal On Advances in Systems and Measurements*, vol. 8, pp. 230–240, 2015.
14. O. Ormandjieva, I. Hussain, and L. Kosseim, "Toward a text classification system for the quality assessment of software requirements written in natural language," in *Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting*, pp. 39–45, 2007.

15. C. Huertas and R. Juárez-Ramírez, "Towards assessing the quality of functional requirements using english/spanish controlled languages and context free grammar," in *Proc. Third International Conference on Digital Information and Communication Technology and its Applications (DICTAP 2013), Ostrava, Czech Republic on*, pp. 234–241, Citeseer, 2013.
16. H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Speculative requirements: Automatic detection of uncertainty in natural language requirements," in *2012 20th IEEE International Requirements Engineering Conference (RE)*, pp. 11–20, IEEE, 2012.
17. E. Parra, C. Dimou, J. Llorens, V. Moreno, and A. Fraga, "A methodology for the classification of quality of requirements using machine learning techniques," *Information and Software Technology*, vol. 67, pp. 180–195, 2015.
18. V. Moreno, G. Génova, E. Parra, and A. Fraga, "Application of machine learning techniques to the flexible assessment and improvement of requirements quality," *SOFTWARE QUALITY JOURNAL*, 2020.
19. ISO, "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering," *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, 2018.
20. G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado, and V. Moreno, "A framework to measure and improve the quality of textual requirements," *Requirements engineering*, vol. 18, no. 1, pp. 25–41, 2013.
21. Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced lstm for natural language inference," *arXiv preprint arXiv:1609.06038*, 2016.
22. A. Ferrari, G. O. Spagnolo, and S. Gnesi, "Pure: A dataset of public requirements documents," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 502–505, IEEE, 2017.
23. Stanford NLP Group *Nltk.org*, 2020.
24. Stanford NLP Group *Nlp.stanford.edu*, 2020.
25. International Council On Systems Engineering (INCOSE), "Guide for writing requirement," *San Diego (CA): INCOSE. 2017. Standard No.: INCOSE-TP-2010-006-02.1*, 2017.
26. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.