ISSN 1870-4069

Generalización de requisitos de software de dominio específico para clasificación de texto

J. Manuel Pérez-Verdejo, Ángel Juan Sánchez-García,
Jorge Octavio Ocharán-Hernández

Universidad Veracruzana,
Facultad de Estadística e Informática,
México

manuel.vrdjo@gmail.com, {angesanchez, jocharan}@uv.mx

Resumen. Los requisitos de software representan un conjunto de características que un sistema debe satisfacer y son críticos para el desarrollo de proyectos de software. El conocimiento que representan tiene una estructura sintáctica definida y un dominio propio de cada proyecto. Sin embargo, este vocabulario específico dificulta su reutilización en otros ámbitos, incluyendo su uso para el entrenamiento de modelos de clasificación en aprendizaje automático. Motivado por estos factores, este estudio experimental propone identificar si la generalización del conocimiento propio de un dominio permite incrementar la precisión de modelos para clasificar requisitos de software en categorías de atributos de calidad. Para comparar los resultados del texto procesado contra el original, se llevó a cabo el entrenamiento de una red neuronal convolucional. Los resultados preliminares muestran que la aplicación de este procedimiento incrementa la precisión del clasificador en un 10 % a comparación del texto original.

Palabras clave: Procesamiento de lenguaje natural, requisitos de software, clasificación, redes neuronales.

Generalization of software requirements in specific domains for text classification

Abstract. Software requirements represent a set of characteristics that a system must satisfy and are critical to the development of software projects. The knowledge they represent has a defined syntactic structure and its own domain of each project. However, this specific vocabulary makes it difficult to reuse it in other areas, including its use for training classification models in machine learning. Motivated by these factors, this experimental study proposes to identify whether the generalization of a specific domain knowledge allows increasing the precision of models to classify software requirements into categories of quality attributes.

To compare the results of the processed text against the original, the training of a convolutional neural network was carried out. Preliminary results show that applying this procedure increases the accuracy of the classifier by 10% compared to the original text.

Keywords: Natural language processing, software requirements, classification, neural networks.

1. Introducción

En el proceso de desarrollo de software, los requisitos representan el eje central de las actividades a realizar en todo el ciclo de vida de un sistema. Es a partir del conocimiento plasmado en estos enunciados que se conoce de forma previa las funcionalidades y especificaciones del software. Sin embargo, el lenguaje natural es ambiguo y frecuentemente interpretable, por lo que se considera una buena práctica el uso plantillas para la redacción de requisitos de calidad [18]. La estructura sintáctica con la que se propone se redacten los requisitos se define en un enunciado del tipo "El [nombre del sistema] deberá ser capaz de [objetivo y respuesta del sistema]" [17].

Adicional a esta estructura definida, los requisitos son redactados en términos del dominio al que pertenecen. Este dominio constituye esencialmente el glosario del proyecto [13]. Los términos específicos de un dominio del conocimiento representan el vocabulario con el que se expresan estos requisitos. De esta forma, se establecen niveles de abstracción que corresponden a qué tan específicos son los requisitos, de entre los que destacan dos niveles: los requisitos específicos de un dominio, y los requisitos genéricos [15].

Este criterio al nivel de detalle también se presenta en una categoría de los requisitos de software conocida como requisitos de calidad. Este tipo de requisitos no describe una función, sino la estructura y el comportamiento con el que se realiza la misma [2]. Estos cuentan con una taxonomía definida [5], y se dividen en categorías como disponibilidad, tolerancia a fallos, mantenibilidad, rendimiento, escalabilidad, seguridad, usabilidad entre otras. Estas características deseables en el software, son conocidas como atributos de calidad.

Sin embargo, el nivel de abstracción con el que se redactan este tipo de enunciados afecta directamente su reusabilidad. Esto es, entre más específico sea el dominio de un requisito, es menos probable que se pueda reutilizar junto con otros de distintos dominios [18]. Esta variabilidad de vocabulario y contexto afecta también en tareas de automatización, principalmente en la clasificación de texto.

Es así como el presente estudio parte del supuesto que al transformar requisitos específicos de un dominio en requisitos genéricos, se puede incrementar la precisión de modelos de clasificación de requisitos, específicamente de atributos de calidad.

El resto del documento se divide de la siguiente forma: en la sección 2 se describen trabajos relacionados en el área de la clasificación de requisitos y diferentes enfoques de preprocesamiento para los mismos.

En la sección 3 se explica la propuesta de generalización empleada en la realización de este trabajo. En la sección 4 se muestran los resultados obtenidos y la comparación de desempeño a través del entrenamiento de un modelo de clasificación con el texto preprocesado y el texto sin ningún tratamiento. Finalmente, la sección 5 resume los hallazgos identificados y el trabajo futuro.

2. Trabajo relacionado

El tópico de la clasificación de requisitos de software ha sido abordado por la literatura con anterioridad dentro de área de Ingeniería de Requisitos. En [6] se prueban diferentes combinaciones de representaciones de texto y clasificadores para identificar de forma automática *app reviews* asociadas con requisitos funcionales o alguna categoría de atributo de calidad (usabilidad, confiabilidad, portabilidad y rendimiento). En este caso se exploran *Bag Of Words*, CHI^2, TF-IDF y *Word2Vec* para la representación del texto y Naïve Bayes, J48 y Bagging para su clasificación.

De la misma forma, el trabajo de Abad et. al. [1] se investiga la clasificación de requisitos utilizando enfoques como modelado de tópicos, *clustering* y clasificación bayesiana para identificar categorías de atributos de calidad. En el caso de este estudio, se realizó la redacción de reglas temporales que limitan el vocabulario complejo propio de este tipo de requisitos. Dado que en este estudio se parte de una base de datos con requisitos provenientes de distintos proyectos de software, se realiza un preprocesado con la intención de reducir la inconsistencia entre los mismos. Adicional a un proceso de etiquetado temporal, se componen reglas de expresiones específicas de un dominio que harían referencia a alguna categoría de atributos de calidad. De esta forma expresiones como *authorization* o *encrypt* se pueden asociar a la categoría de *Security*.

Otros estudios, centran su investigación en la identificación de un sólo tipo de atributo de calidad. En [11] se propone el uso de redes neuronales convolucionales para identificar reportes de fallas asociados con el atributo de calidad de Seguridad. En este estudio, se probaron diferentes configuraciones de redes neuronales y se realizó el entrenamiento con texto recolectado de repositorios de GitHub, artículos de Wikipedia y documentación propia de una empresa. Para el caso del preprocesamiento de su texto, únicamente se realizó la vectorización de sus documentos utilizando la biblioteca de *Word2Vec*.

Por otra parte, el estudio llevado a cabo por Rago, Marcos y Diaz-Pace [12] realiza una comparación de diferentes combinaciones semánticas, que permiten enriquecer la información provista en los requisitos de software. De esta forma, se usa la información dentro del dominio de los requisitos, para asociar conceptos que podrían no tener el mismo significado en otro dominio. Tal es el caso de palabras como *register* y *save*, que en el contexto de área de la Ingeniería de Software podrían referirse al mismo concepto. Su método de enriquecimiento semántico demostró una mejora de hasta un 18% en la precisión y exhaustividad de la clasificación de requisitos.

3. Propuesta

El procedimiento de generalización de dominio de los requisitos consiste en la aplicación de tres fases de etiquetado y sustitución de expresiones específicas, las cuales se muestran en la Figura 1. Para la ejecución de la propuesta se identificaron dos fuentes de ejemplos de requisitos de software. Ambos conjuntos fueron preprocesados y comparados con sus respectivas versiones originales.

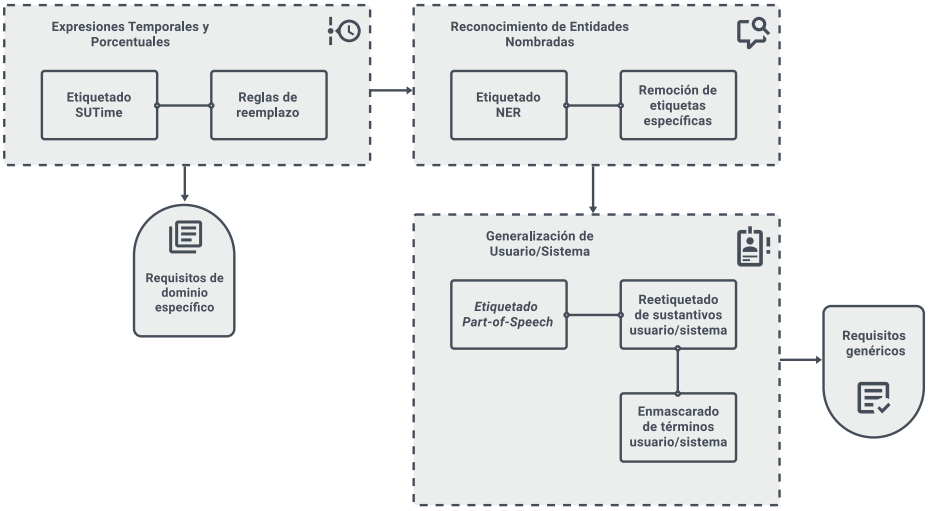

Fig. 1. Procedimiento de generalización del dominio.

3.1. Conjuntos de datos

El entrenamiento se realizó con una muestra de la base de datos de requisitos de software PROMISE [14]. Este conjunto incluye un total de 625 diferentes ejemplos de requisitos etiquetados y provenientes de 15 diferentes proyectos. Se recolectaron todos aquellos requisitos asociados a atributos de calidad. De esta forma, se identificaron 256 ejemplos de las siguientes categorías:

- Disponibilidad,
- Tolerancia a fallos,
- Mantenibilidad,
- Rendimiento,
- Escalabilidad,
- Seguridad,
- Usabilidad.

Por otra parte, para probar la precisión del modelo, se utilizaron 40 ejemplos de atributos de calidad del libro *The Quest for Software Requirements* [8]. Este conjunto de enunciados no son de proyectos en común y provienen de dominios diferentes. Únicamente se seleccionaron de las siguientes categorías que coinciden con enfoque del estudio:

- Disponibilidad,
- Mantenibilidad,
- Escalabilidad,
- Seguridad,
- Usabilidad.

3.2. Expresiones temporales y porcentuales

Con la versión en texto plano de los requisitos, se realizó un etiquetado automático en tres fases utilizando la biblioteca Stanford CoreNLP [7]. Como fase inicial, se identificaron todas las expresiones temporales (intervalos de tiempo, fechas, duraciones, etc.) utilizando la biblioteca de SUTime. [3].

Con la etiquetas generadas, se aplica una versión extendida de las reglas temporales propuestas por Abad et. al. [1] y se agregan expresiones porcentuales:

- Primera Regla: Aquellas expresiones etiquetadas como *"DURATION"* y que incluyan alguna de las siguientes expresiones serán sustituidas por *within*: *no longer than, in under, no more than, not be more than, no later, in, for less than, at a maximun.*
- Segunda Regla: Las expresiones *24 hours per day 365 days per year, 24 hour* se sustituyen por *alltimes.*
- Tercera Regla: Las expresiones de tipo *"DURATION"* y que incluyen *within* y un lapso de tiempo en minutos o segundos, se sustituyen por *fast.*
- Cuarta Regla: Las expresiones que incluyen *timely* o *quick* se sustituyen por *fast.*
- Quinta Regla: Las expresiones que contienen un porcentaje mayor a 80% y la frase *of the time* se sustituyen por *alltimes.* Aquellas que contengan un porcentaje mayor a 80% y la frase *up time* se sustituyen por *uninterrupted uptime.* Las expresiones restantes, si son de tipo *"IN"* o *"SET"* se sustituyen por *alltimes.*
- Sexta Regla: Los porcentajes iguales a 100% se sustituyen por *all*, los mayores a 50% y menores a 100% se sustituyen por *most.* Los porcentajes restantes se transforman en la expresión *little.*

Durante la ejecución de estas reglas, se pudo identificar que las categorías de atributos de calidad que más presentan expresiones de este tipo, son las de rendimiento y de disponibilidad. De esta forma, el siguiente requisito referente al atributo de calidad de rendimiento:

"The CMA report shall be returned no later 60 seconds after the user has entered the CMA report criteria"

Es transformado en la siguiente expresión, que conserva el interés en la velocidad del sistema, pero límita el detalle del tiempo en específico:

"The CMA report shall be returned fast after the user has entered the CMA report criteria"

3.3. Reconocimiento de entidades nombradas

El corpus resultante de la transformación de expresiones temporales se etiqueta utilizando el modelo NER de Stanford CoreNLP [4]. Es de esta forma que se identifican entidades que son dependientes del dominio de los requisitos. Para reducir la complejidad de estos del vocabulario, se eliminan aquellas palabras asociada con alguna de las siguientes etiquetas: *NATIONALITY, NUMBER, ORGANIZATION, COUNTRY, TIME, STATE_OR_PROVINCE.* Así, retomando el ejemplo anterior, se retirarían las siglas *CMA*, detectadas como una organización, resultando en la siguiente expresión genérica:

"The report shall be returned fast after the user has entered the report criteria"

3.4. Generalización de usuario/sistema

Finalmente se realizó un etiquetado utilizando la biblioteca de *Part-of-Speech* [16]. Este procedimiento identifica la categoría léxica que ocupa una palabra dentro de una oración (verbos, sustantivos, adjetivos, por mencionar algunos). Con las etiquetas generadas, se llevó a cabo una revisión utilizando la herramienta para anotación de texto doccano [10]. Es así como, de manera manual, se realizó el reetiquetado en el que se identificó el nombre del sistema y el nombre del usuario únicamente considerando aquellas palabras o frases que ya habían sido etiquetadas como sustantivos previamente. Con las etiquetas usuario/sistema se procedió a enmascarar esta información. Por ejemplo:

"The clinical site shall be intuitive to the Program Administrators / Nursing Staff Members"

Se reduce al enunciado:

"The system shall be intuitive to the user"

4. Experimentos y resultados

Para la evaluación de la propuesta, se realizó el entrenamiento de dos modelos de clasificación de texto utilizando la biblioteca de *deep learnig* de Ludwig [9]. Un modelo se entrenó y probó con los requisitos en su forma original, y otro con los requisitos en nivel de abstracción genérico. La características para le generación del modelo incluyen un proceso de remoción de *stopwords* y de *lemmatization.* Se usa la arquitectura de una red neuronal convolucional con el *encoder* de *parallel_cnn* incluido en la biblioteca. El entrenamiento se realiza utilizando los parámetros por defecto de 100 *epoch* y un *batch_size* de 128. La Figura 2 resume este proceso.

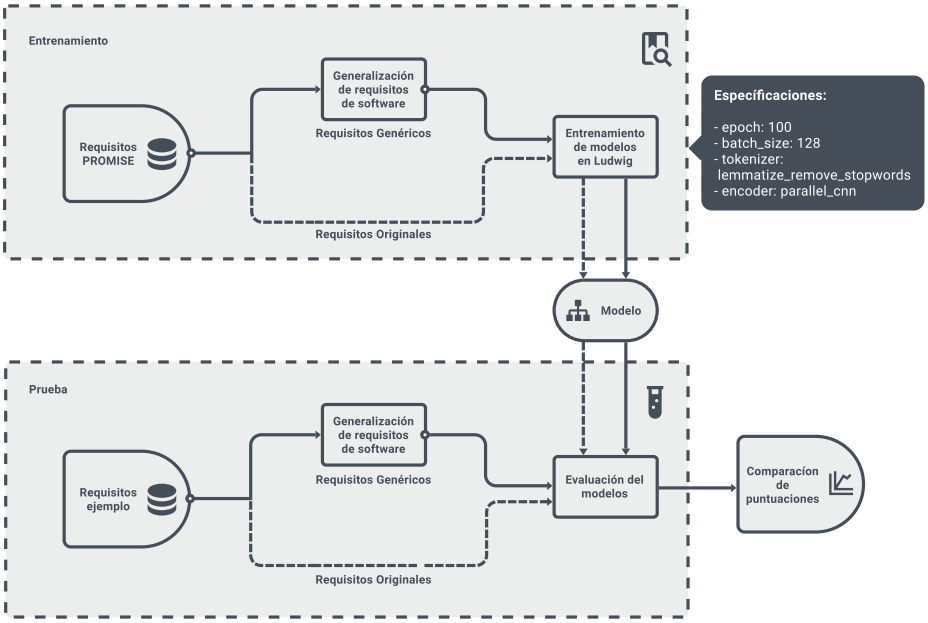

Fig. 2. Procedimiento experimental para la evaluación de la propuesta.

La comparación del desempeño de ambos modelos se muestran en la Figura 3. Se puede observar una mayor tendencia incremental por parte del modelo entrenado con requisitos genéricos, en términos de la precisión con el conjunto de prueba.

El modelo entrenado usando requisitos de dominio específico muestra evidencia de incompatibilidad al momento de predecir requisitos que están fuera de sus dominios conocidos, consiguiendo una precisión de 42.5%. De esta forma, el procedimiento de generalización logró incrementar la precisión de la clasificación en un 10%. La información detallada de ambas ejecuciones se puede consultar en el repositorio del experimento[1], así como el código y las bases de datos[2].

Adicionalmente, se realizó la matriz de confusión de ambos modelos. Se puede observar cómo el modelo entrenado con requisitos genéricos reduce la clasificación errónea de las categorías de Seguridad y de Rendimiento, sin embargo esta última no formaba parte del conjunto de prueba. A pesar de ello, ambos modelos fallaron en clasificar correctamente los atributos de calidad de Mantenibilidad, categoría que sí se encontraba incluida en el conjunto de prueba.

La reducción del error se puede observar en mayor medida en la Figura 5. Mientras que los valores de precisión y valor-F de las categorías de Disponibilidad y Seguridad incrementan considerablemente, la categoría de Usabilidad experimenta un decremento. Asḿismo, es posible observar que el procedimiento de

[1] https://comet.ml/manolomon/domain-masking
[2] http://github.com/quality-attributes/domain-masking

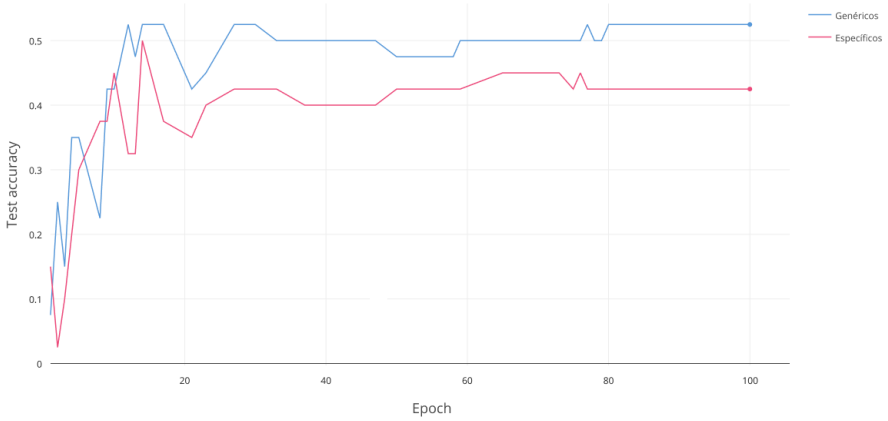

Fig. 3. Evolución a través de los ciclos (*epoch*) de la precisión del modelo en términos del conjunto de prueba.

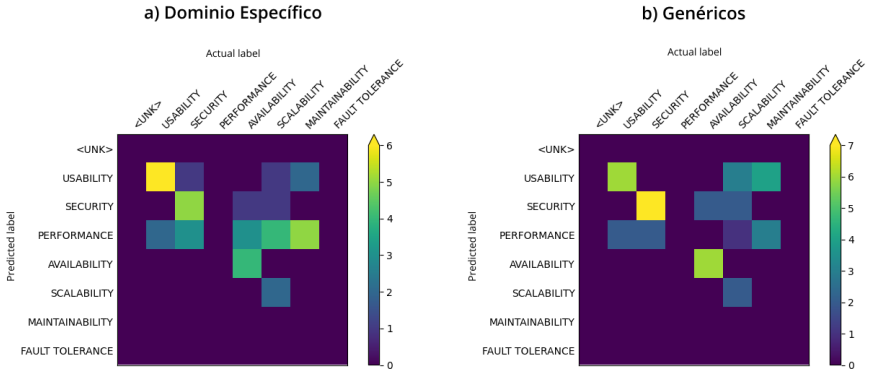

Fig. 4. Matrices de confusión de los modelos sin el procesamiento y una vez aplicado el tratamiento propuesto.

generalización no representó ningún cambio en las puntuaciones de la categoría de Escalabilidad.

5. Conclusiones y trabajo futuro

La clasificación automática de requisitos de software representa un campo de oportunidad para la colaboración de la Ingeniería de Requisitos con la Inteligencia Artificial.

Estos artefactos son críticos para el desarrollo de sistemas, y especialmente en proyectos de gran escala, representan inversiones importantes de dinero y

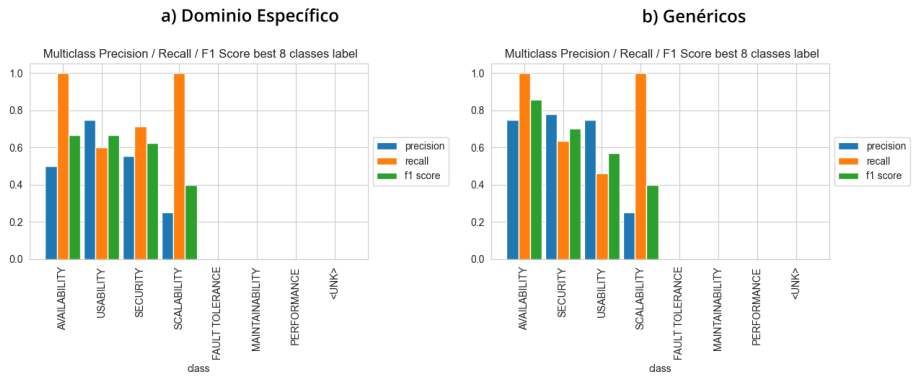

Fig. 5. Precisión, exhaustividad y valor-F de cada categoría evaluada por los modelos.

esfuerzo. Identificar una forma de reutilizar el conocimiento plasmado en los requisitos de forma automática significa una oportunidad para dar paso a algoritmos de clasificación de requisitos con menor margen de error.

Así, en este artículo se presentó una evaluación experimental con el objetivo de investigar un posible enfoque para la transformación de requisitos específicos de un dominio en requisitos genéricos. Los resultados iniciales muestran un incremento en la precisión en un 10% a comparación con el texto sin ningún tratamiento. Sin embargo, esta precisión permanece siendo baja (52.5%), por lo que se contempla explorar campos adicionales del área de estudio del Procesamiento de Lenguaje Natural, así como otras alternativas de clasificadores.

Por otra parte, se identificó que este tipo de tratamiento beneficia la clasificación acertada de ciertas categorías como el Rendimiento y la Seguridad, pero reduce la exhaustividad de otras clases como la Usabilidad.

Una vez analizados los resultados, se identifica cabida para la aplicación de técnicas adicionales, como el uso de palabras clave para agregar pesos modificados, o el uso de mejoras sintácticas. Por otro lado, tanto el entrenamiento y las pruebas de los modelos se vieron limitados por la cantidad de ejemplos disponibles, por lo que se considera la adquisición de más ejemplos documentados.

Finalmente, la fase de enmascaramiento de usuario/sistema se realizó de forma manual, convirtiendo el procedimiento propuesto en semiautomático. No obstante, se identificó como una posibilidad viable para futuros estudios, el entrenamiento de modelos de NER para ocultar esta información de forma automática.

References

1. Abad, Z.S.H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., Schneider, K.: What Works Better? A Study of Classifying Requirements. In: 2017 IEEE 25th International Requirements Engineering Conference (RE). pp. 496–501. IEEE (sep 2017)

2. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley Professional, 3rd edn. (2012)
3. Chang, A.X., Manning, C.: SUTime: A library for recognizing and normalizing time expressions. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 12). pp. 3735–3740. European Language Resources Association (ELRA), Istanbul, Turkey (2012)
4. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by Gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05. pp. 363–370. ACL '05, Association for Computational Linguistics, Morristown, NJ, USA (2005)
5. ISO/IEC/IEEE: ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering. ISO/IEC/IEEE 29148:2018(E) pp. 1–104 (2018)
6. Lu, M., Liang, P.: Automatic Classification of Non-Functional Requirements from Augmented App User Reviews. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. pp. 344–353. EASE'17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3084226.3084241
7. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The Stanford CoreNLP Natural Language Processing Toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 55–60. Association for Computational Linguistics, Stroudsburg, PA, USA (2014)
8. Miller, R.E.: The Quest for Software Requirements. MavenMark Books, Oconomowoc, WI, USA (2009)
9. Molino, P., Dudin, Y., Miryala, S.S.: Ludwig: a type-based declarative deep learning toolbox (2019)
10. Nakayama, H., Kubo, T., Kamura, J., Taniguchi, Y., Liang, X.: doccano: Text Annotation Tool for Human (2018), https://github.com/doccano/doccano
11. Palacio, D.N., McCrystal, D., Moran, K., Bernal-Cardenas, C., Poshyvanyk, D., Shenefiel, C.: Learning to Identify Security-Related Issues Using Convolutional Neural Networks. In: Proceedings - 2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019. pp. 140–144 (2019)
12. Rago, A., Marcos, C., Diaz-Pace, J.A.: Using semantic roles to improve text classification in the requirements domain. Language Resources and Evaluation 52(3), 801–837 (2018)
13. Rosenberg, D., Stephens, M.: Use Case Driven Object Modeling with UML. Apress, Berkeley, CA (2007)
14. Sayyad Shirabad, J., Menzies, T.J.: The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada (2005), http://promise.site.uottawa.ca/SERepository
15. Shehata, M.S., Eberlei, A., Hoover, H.J.: Requirements Reuse and Feature Interaction Management. In: Proceedings of the 15th International Conference on Software & Systems Engineering and their Applications. Paris (2002)
16. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics. EMNLP '00, vol. 13, pp. 63–70. Association for Computational Linguistics, Morristown, NJ, USA (2000)

17. Wiegers, K.E.: More About Software Requirements: Thorny Issues and Practical Advice. Microsoft Press, USA (2005)
18. Wiegers, K.E., Beatty, J.: Software Requirements. Microsoft Press, USA, 3rd edn. (2013)