Procesamiento de lenguaje natural en la construcción de un modelo léxico

Natalia Mercedes Vidal Monge Navarro

Universidad de Belgrano, Buenos Aires, Argentina
natalia.vidal@comunidad.ub.edu.ar

Abstract. En la Ingeniería de Software, la Ingeniería de Requisitos se centra en el proceso de recopilar y definir los requisitos de un software a desarrollar que cumpla las necesidades de los clientes. Sin embargo, no siempre se logra, siendo una causa importante de ello el no alcanzar el nivel necesario de comprensión del problema, y esto proviene en gran medida de una comunicación fallida entre las partes involucradas. Este estudio se centra en el modelo Léxico Extendido del Lenguaje, que desempeña un papel fundamental en la comunicación entre las partes en la definición de requisitos y en la construcción de otros modelos. Actualmente, la formulación manual del modelo conlleva inconvenientes como el tiempo invertido y la introducción de sesgos del ingeniero de requisitos. Por ello se propone desarrollar un prototipo basado en Procesamiento de Lenguaje Natural que permita la construcción automatizada del modelo Léxico a partir de una fuente de información textual. En este estudio se exploran técnicas y librerías relevantes, se diseña y desarrolla el prototipo, y se evalúa su eficiencia y eficacia.

Keywords: Procesamiento de Lenguaje Natural, Ingeniería de requisitos, Modelado en Lenguaje Natural, Léxico Extendido del Lenguaje.

1 Introducción

El proceso de construcción de requisitos resulta clave en el marco de la Ingeniería de Software ya que en él se definen los requisitos del software a ser construido en base a información del contexto de aplicación. Un entendimiento del problema incompleto o sesgado suele derivar en una definición incorrecta de requisitos, lo que implica gastos de tiempo, monetarios y/o errores no detectados oportunamente, que suelen encadenar sucesivos errores y gastos [1-2]. Este proceso de requisitos entonces exige una eficaz comunicación entre las partes, la que se dificulta por el uso de lenguajes diferentes entre los involucrados.

La Ingeniería de Requisitos (IR) propone modelos útiles para el proceso de construcción de requisitos usando lenguaje natural, entre ellos, el Léxico Extendido del Lenguaje (LEL) [3], que define el vocabulario relevante en el contexto de aplicación, resultando de importancia y utilidad para la comunicación entre las partes y para la construcción de otros modelos [1]. La creación de un LEL con calidad es clave y justifica la búsqueda de nuevas técnicas y herramientas a aplicar para mejorar su calidad y su

velocidad de construcción.

La estrategia actualmente empleada para la formulación de un LEL es realizada de forma iterativa por ingenieros de requisitos familiarizados con el contexto a través de las fuentes de información identificadas [4], implicando una alta demanda de tiempo y posibles sesgos y subjetividades provenientes de los ingenieros [5-6].

Se considera, en base a estudios anteriores [7-8], que las técnicas de Procesamiento de Lenguaje Natural (PLN) aplicadas a la construcción del LEL pueden mejorar estos inconvenientes. Existen múltiples técnicas y librerías de PLN que pueden aplicarse al tratar modelos en lenguaje natural en la IR.

En este trabajo se presenta el desarrollo de un prototipo de construcción de LEL para optimizar su generación aplicando técnicas de PLN, buscando reducir el tiempo requerido y evitar las subjetividades y los errores provenientes de la intervención humana. Este trabajo es parte del trabajo final de carrera "Ing. de Requerimientos: modelo de procesamiento de lenguaje natural para construcción del Léxico Extendido del Lenguaje" desarrollado en el marco de la carrera de Ingeniería en Informática de la Universidad de Belgrano, siendo la tutora la Dra. Graciela Hadad.

Se describe en las siguientes secciones el modelo LEL, la selección de librerías y técnicas de PLN a utilizar, la estrategia de construcción, el diseño e implementación del prototipo y la mejora introducida, para finalmente exponer conclusiones.

2 Modelo Léxico Extendido del Lenguaje

Dado que en el contexto de aplicación del software los clientes poseen un lenguaje propio, la comprensión del lenguaje resulta imperativa para los ingenieros, permitiendo una comunicación efectiva entre las partes a lo largo de todo el proceso de IR [3, 1]. Debido a esto, resulta conveniente como primera actividad del proceso la creación de un LEL.

El LEL, como "una técnica para especificar el conocimiento del contexto de aplicación" [1], describe en lenguaje natural los términos relevantes en ese contexto, asentando conocimiento lingüístico útil para el desarrollo de software [3].

2.1 Estructura del Modelo

En el LEL un *símbolo* o término, tiene nombre, noción (denotación del símbolo) e impacto (connotación del símbolo); y pertenece a uno de cuatro tipos: *sujeto* realiza acciones, *objeto* es un elemento sobre el cual un sujeto actúa, *verbo* es una acción que un sujeto realiza posiblemente sobre un objeto, y *estado* es una situación en la que se encuentran sujetos, objetos o verbos.

Para obtener un modelo LEL de mejor calidad la descripción de símbolos debe seguir los principios de circularidad y vocabulario mínimo, en base a los cuales debe utilizarse la mayor cantidad de símbolos pertenecientes al LEL al definir un símbolo y con la menor cantidad posible de vocabulario externo.

Los símbolos pueden estar relacionados entre sí como sinónimos que se unen en un solo símbolo con múltiples nombres, u homónimos con igual nombre, pero diferente

noción e impacto. También pueden estar en jerarquías taxonómicas o mereológicas, donde hay símbolos genéricos y especializados, o donde son parte de una entidad mayor.

2.2. Construcción Manual Iterativa del LEL

La estrategia actual de construcción iterativa del LEL tiene como base que el ingeniero de requisitos identifique, defina y evalúe potenciales símbolos a lo largo de múltiples iteraciones, tratando en cada iteración un conjunto acotado de símbolos [4-5] (ver Fig. 1).

Se utiliza como entrada al proceso de construcción información textual proveniente de la transcripción de entrevistas y reuniones o de documentación brindada por los clientes u otro material de elicitación elaborado en formato texto. La identificación y descripción de símbolos determina los *símbolos candidatos* a estar en el modelo. En la primera iteración el ingeniero identifica un conjunto inicial de términos *semilla* que se toman como símbolos candidatos. De estos se seleccionan hasta seis símbolos con la mayor importancia percibida y se inicia su clasificación y descripción.

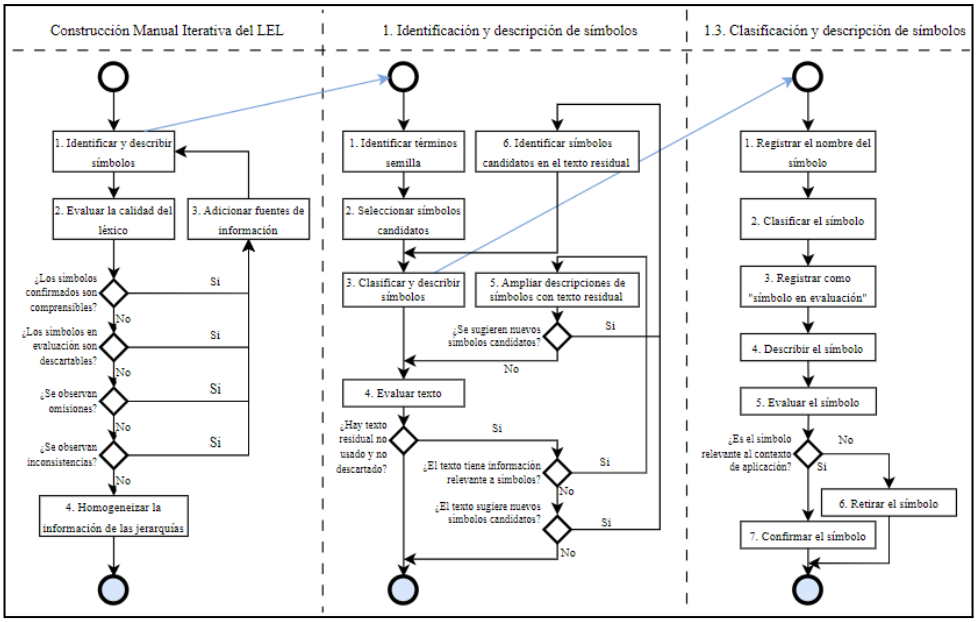

Fig. 1. Proceso simplificado de Construcción Iterativa del LEL. Fuente: Autora.

Para cada símbolo se *registra su nombre en el LEL* y se lo *clasifica s*egún su tipo, considerando la posible existencia de relaciones con otros símbolos. Ya clasificado se lo *registra como un símbolo en evaluación*. Luego, para su *descripción* se adiciona su noción e impacto. Cada instancia del símbolo en el texto fuente se marca según haya sido usada y

se busca nueva información a agregar a símbolos y que proporcione nuevos posibles símbolos candidatos.

Se *evalúa el símbolo:* debe tener nombre, noción, impacto y relevancia en el contexto de la aplicación. De no cumplir se *retira el símbolo del LEL*. De cumplir, se *confirma el símbolo*, se marcan todas sus ocurrencias en el LEL y se buscan relaciones con los demás símbolos confirmados e información inconsistente.

Se *evalúa el texto* fuente restante y de poder utilizarse para ampliar símbolos o de identificarse nuevos símbolos candidatos se reiteran las actividades correspondientes.

En la *evaluación de la calidad del léxico*, se determina que el léxico construido es de calidad aceptable si los símbolos tienen definiciones completas y son de utilidad para el entendimiento del lenguaje. De ser satisfactoria, se *homogeneiza la información* en las jerarquías detectadas de símbolos y se da como finalizado el modelo. Sino se retoman las actividades de adicionar fuentes de texto, y de identificar y describir símbolos.

3 Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (PLN) es un subcampo dentro de la Inteligencia Artificial [9][10] referente a la interacción entre la computadora y el lenguaje natural. Para la construcción del prototipo se utilizaron las librerías NLTK [11], PKE [12] y spaCy [13], que soportan el uso de las técnicas detalladas a continuación, en Google Collaboratory [14], que proporciona un entorno de desarrollo y simplifica el proceso de instalación de las librerías utilizadas.

Se utilizaron las siguientes técnicas:

- Tokenización: segmenta el texto procesado en unidades menores (*tokens)* relevantes para el análisis del texto [15].
- Lematización: obtener el lema, la versión base, de una palabra [16].
- StopWords: definición de palabras irrelevantes para el análisis del texto [17].
- TextRank: procesa un texto como un grafo con nodos de tokens con un mismo puntaje inicial de importancia, relacionados por su co-ocurrencia [18][19]. Los puntajes se recalculan iterativamente según la importancia de los vecinos y el peso de las relaciones con ellos. Tomando un valor T de un tercio de los vértices totales con los mayores puntajes se permite la definición de *keyphrases* del texto.
- Part-of-Speech (POS) Tagging: etiqueta tokens por su parte gramatical en el texto [20].
- Dependency (DEP) Parsing: determina las relaciones sintácticas entre los tokens de una oración, construyendo un árbol de dependencia representando la estructura sintáctica de la oración. Los nodos *heads* tienen palabras que dependen de ellas y son modificables por los nodos *dependents*.[21]. Las dependencias en la oración nacen de una palabra *root*. La librería spaCy usa las anotaciones Dependencias Universales para representar con etiquetas las relaciones entre *heads* y *dependents* [22].
- Named Entity Recognition: reconocimiento de Named Entities (NEs), objetos designables con un nombre propio [23].

4 Prototipo de Procesamiento de Lenguaje Natural

El prototipo se ejecuta mediante los siguientes pasos: carga del texto fuente; detección de potenciales símbolos; limpieza de símbolos; iniciación y nombramiento de símbolos; clasificación; descripción de noción e impacto; referenciación de símbolos; detección de sinónimos; detección de jerarquías entre símbolos; configuración de nombres; e impresión del LEL. A continuación, se resumen estos pasos, que se detallan en [24].

El texto a partir del cual se construye el LEL se carga como un archivo de tipo txt al Google drive, se accede con PlaintextCorpusReader y se define como el corpus a procesar.

Se establece el corpus procesado por el modelo spaCy es_core_news_sm como fuente y se le extraen keyphrases con TextRank, sin considerar las stopwords. Se limitaron las keyphrases al valor T calculado y se definen como símbolos potenciales del LEL.

Finalmente, si varios tokens en el nombre de un símbolo tienen DEP tag de tipo objeto, se crean nuevos símbolos potenciales con estos tokens y sus dependientes, eliminando al original, y se quitan de los nombres de símbolos los tokens con stopwords como lema.

4.1 Armado de símbolos

Para el tratamiento de los símbolos se define una clase ***Simbolo*** con los atributos:

- *nombre*: string correspondiente al nombre del símbolo
- *clase*: string correspondiente al tipo del símbolo.
- *nocion*: lista de strings de oraciones que componen la noción del símbolo.
- *impacto*: lista de strings de oraciones que componen el impacto del símbolo.
- *otros_nombres*: lista de strings de otros nombres (sinónimos) del símbolo.
- *especializaciones*: lista de sus símbolos especializados en una jerarquía.
- *es_referenciado*: boolean según sea el símbolo referenciado por otros.
- *hace_referencia*: boolean según haga referencia el símbolo a otros.

Se crea la lista de objetos *Simbolo*, denominada *simbolos_confirmados*, en la que se inicializan los símbolos potenciales y se definen sus tipos: de ser una NE, el símbolo es catalogado de tipo Sujeto; de no serlo, se lo clasifica en base a POS tags y DEP tags.

Por POS tags, si la tag es de verbo, sustantivo o adjetivo entonces se asigna el tipo Verbo, Sujeto o Estado, respectivamente. De ser la tag de pronombre o artículo, es de tipo Objeto. Si no aplica en los casos anteriores, el tipo es nulo.

Por DEP tags, si la tag refiere a un sujeto nominal (*nsubj* o *nsubjpass)* es de tipo Sujeto. Si la tag refiere a objetos sintácticos, es de tipo Objeto. Si refiere a verbos (*root*), complementos o modificadores de un predicado (*ccomp* o *advcl*), es de tipo Verbo [22]. Si se refiere a adjetivos o adverbios, es de tipo Estado. De ser otro el caso, es de tipo nulo.

Finalmente, se toma el tipo definido por ambas clasificaciones en caso de coincidir o de ser alguno nulo, priorizando el no nulo. De diferir los tipos sin ser nulos, como consecuencia de la observación empírica se prioriza la clasificación por DEP tags siempre que esta no lo defina como Verbo, donde se prioriza la clasificación por POS tags y si esta apunta a un tipo sujeto se lo toma como objeto.

Para la descripción de los símbolos, si el token *root* del nombre del símbolo en una

oración en el texto fuente es de tipo Sujeto o Estado según DEP tags, entonces se agrega la oración a la noción; en su defecto, se agrega al impacto. Si un símbolo tiene noción e impacto vacíos se lo quita de la lista *símbolos_confirmados*.

4.2 Configuración de símbolos

Las referencias entre símbolos se marcan haciendo uso de los atributos *hace_referencia* y *es_referenciado*; y se quitan los símbolos que no tienen relación con el resto del LEL.

Se marcan las relaciones entre símbolos sinónimos y jerarquías taxonómicas. Los sinónimos se toman como redundancias con igual tipo, noción e impacto; y se asimilan en un solo símbolo. Las jerarquías taxonómicas se marcan entre dos símbolos si un primer símbolo tiene como nombre la *root* del nombre de un segundo, su especialización.

En cuanto a los nombres de los símbolos, deben utilizar la forma singular y la forma infinitiva en caso de verbo [3]. Para cada nombre, con la excepción de los símbolos del tipo estado, se reemplazan los tokens que no cumplan con esto por su lema.

Por último, se presentan los símbolos para descargar el LEL en formato .txt.

5 Evaluación del Prototipo

Se evaluó la efectividad del prototipo en la construcción del LEL a través del grado de coincidencia en símbolos, su clasificación y descripción, y costo temporal de realización del LEL. Se realizó la evaluación con dos casos de complejidad media, sobre un sistema para uso de bicicletas en estaciones en una ciudad y otra para evaluación de desempeño laboral. Si bien se aseguró la calidad de los LELs construidos manualmente, se debe tener presente la posible inclusión de sesgos, ambigüedades u otros errores humanos en ellos.

Se construyeron LELs siguiendo el proceso iterativo manual y utilizando el prototipo desarrollado para los dos casos y se obtuvieron los resultados de la Tabla 1 para cada caso.

Tabla 1: Resumen de resultados de LELs

Casos:	Caso 1: EcoBici		Caso 2: Evaluación de Desempeño	
Proyecto:	LELmanual	LELprototipo	LELmanual	LELprototipo
Total de símbolos	22	59	19	31
Total de símbolos de tipo SUJETO	5	0	5	4
Total de símbolos de tipo OBJETO	8	49	6	25
Total de símbolos de tipo VERBO	8	1	8	1
Total de símbolos de tipo ESTADO	1	9	0	1
Total de nociones	43	27	27	18
Total de impactos	110	121	57	82
Promedio de nociones por símbolo	1,95	0,46	1,42	0,58
Promedio de impactos por símbolo	5,00	2,05	3,00	2,65

Para la evaluación de la coincidencia en la identificación y nombramiento de los símbolos se tomaron las siguientes medidas, cuyos resultados se presentan en la Tabla 2.

- Grado de paridad: coincidencia entre las cantidades totales de símbolos como el total de símbolos en el LELmanual sobre el total de símbolos en el LELprototipo.
- Grado de equivalencias en el otro LEL: el porcentaje de símbolos del LEL que tienen símbolos semejantes en el otro LEL.

Tabla 2: Resultados de evaluación de símbolos

	Caso 1: EcoBici		Caso 2: Evaluación de Desempeño	
Versiones:	LELmanual	LELprototipo	LELmanual	LELprototipo
Grado de paridad del total	37,29%		61,29%	
Total de símbolos con equivalencias en el otro LEL	5	8	9	15
Grado de equivalencias en el otro LEL	22,73%	13,56%	47,37%	48,39%

El grado promedio de símbolos con equivalencias en el otro LEL es de 18,14% para el Caso 1 y de 47,88% para el Caso 2, ambos menores al 50%. La diferencia entre estos porcentajes lleva a considerar que la técnica utilizada en el prototipo para obtener las palabras clave (*keywords*), tomadas como símbolos, aplica un criterio para su selección que no coincide con el del ingeniero.

En el Caso 1 el porcentaje de coincidencia en la clasificación de los símbolos equivalentes fue del 100% y en el Caso 2 fue del 44,44%. La diferencia entre ambos porcentuales, así como el bajo porcentaje obtenido para el Caso 2, evidencia que ninguna de ambas técnicas, ni la combinación entre ellas, termina de ser del todo compatible con la clasificación de símbolos en el LEL. Los métodos para obtener palabras claves del texto, al ser tomadas como símbolos, no las consideran desde el mismo punto de vista que el ingeniero al momento de realizar la captación de los símbolos. Asimismo, la forma en la que se considera la clasificación de las palabras dentro del LEL no coincide con los criterios de clasificación de palabras por etiquetación por POS o DEP tags.

Comparando el LEL por prototipo y LEL manual para cada caso, la estructuración de la *noción* y el *impacto* difirieron. Por prototipo la descripción se estructuró citando oraciones de la fuente. En cambio, en la construcción manual se la estructuró como conclusiones del ingeniero sobre la noción y el impacto del símbolo, lo que puede introducir sesgos del propio ingeniero de requisitos, que se alejen de la información elicitada [6].

En función de las diferencias obtenidas entre el LEL manual y el LEL del prototipo para ambos casos, se consideró el posible uso del prototipo como base para obtener una versión mejorada del LEL, donde el ingeniero aplica correcciones sobre el LEL del prototipo. Es decir, la alternativa sería la construcción del LEL de una manera híbrida, utilizando PLN para una versión inicial y luego realizar ajustes manuales.

Finalmente, para la medición de tiempos se definieron las siguientes métricas, adicionando el posible desarrollo híbrido:

- Tiempo de Ejecución del prototipo desde el procesamiento de la entrada (fuente textual) hasta obtener como salida el LEL.

- Tiempo de Correcciones manuales del ingeniero para completar un LEL teniendo como base el LEL del prototipo.
- Tiempo de Desarrollo manual de un LEL usando el proceso iterativo manual.
- Tiempo de Desarrollo híbrido, sumando el tiempo de ejecución del prototipo y el tiempo de correcciones del ingeniero.

El tiempo de correcciones (t_C) se estimará utilizando el tiempo promedio de desarrollo de un símbolo en la construcción manual (t_{pCM}) y el tiempo promedio de descripción manual de un símbolo (t_{pdCM}). El primero incluye el tiempo total dedicado a la *identificación* de símbolos, su *clasificación* y *descripción* y se calcula en función del tiempo total de desarrollo de la construcción manual (t_{TCM}) y el total de símbolos del LEL manual (sa) (ver fórmula 1), mientras que el segundo involucra solo el tiempo de *descripción* de los símbolos en la construcción manual (t_{dCM}) y el total de símbolos del LEL manual (ver fórmula 2).

$$t_{pCM} = t_{TCM}/sa \tag{1}$$

$$t_{pdCM} = t_{dCM}/sa \tag{2}$$

En el caso de construcción híbrida, el sa está compuesto por símbolos con equivalencias (s_{CE}) (presentes en ambos LELs) y sin equivalencias (s_{SE}) (solo captados manualmente). La identificación y clasificación de símbolos con equivalencias se realiza con el prototipo y se considera despreciable el tiempo dedicado a la posible corrección de su clasificación en la construcción híbrida.

Asumiendo la peor situación, en que la noción e impacto producida por el prototipo para símbolos con equivalencias solamente reduce en un 10% el tiempo dedicado a su descripción, se definió el tiempo teórico de correcciones (t_C) como:

$$t_C = [t_{pdCM} * 0.90 * s_{CE}] + [t_{pCM} * s_{SE}] \tag{3}$$

Finalmente, para las métricas de tiempo definidas se obtuvieron los resultados plasmados en la Tabla 4. Con estos valores se refleja que utilizando el desarrollo híbrido en vez del desarrollo manual se reduce, en el peor caso, en un 6.76% el tiempo requerido para el desarrollo del LEL en el Caso 1 y en un 13.57% en el Caso 2.

Tabla 4: Resultados de medición de tiempos

Casos:	CASO 1	CASO 2
Proyecto:	EcoBici	Evaluación de Desempeño
Tiempo de ejecución del prototipo	0h 2m 28s	0h 2m 2s
Tiempo teórico de correcciones en la peor situación	9h 48m 42s	7h 58m 30s
Tiempo teórico de desarrollo híbrido en la peor situación	9h 51m 10s	8h 0m 32s
Tiempo de desarrollo manual	10h 34m	9h 16m

6 Conclusión

Ante un caso puntual con una fuente textual de mediana complejidad se propuso construir un modelo LEL mediante un prototipo basado en técnicas de procesamiento de lenguaje natural. Para esto, se estableció un marco teórico de la construcción del modelo LEL y de librerías y técnicas de PLN. Posteriormente, se diseñó y programó el prototipo, definiendo la estrategia de construcción del LEL por prototipo.

Al ejecutar la evaluación comparativa entre LELs, se concluye que de la definición de símbolos, el porcentaje promedio de símbolos con equivalencias es menor al 50% entre los LELs realizados, y que la clasificación de los símbolos equivalentes evidencia que ninguna de las técnicas de clasificación de palabras es enteramente compatible con la clasificación de símbolos establecida [25].

El LEL construido por prototipo puede constituirse como una base de datos previa, brindando un soporte inicial al ingeniero para la descripción de los símbolos, redundando en el ahorro de tiempo. Este LEL resultante colaboraría con el *síndrome de la página en blanco*. Usando un desarrollo híbrido compuesto por la ejecución del prototipo y la posterior corrección del LEL producido por prototipo para obtener un LEL final, se reduce en el peor de los casos en casi un 7% el tiempo requerido para el desarrollo del LEL basado en su aplicación en dos casos de estudio.

En base a la reducción del tiempo requerido del ingeniero, se propone hacer uso de técnicas provenientes de la construcción iterativa manual en conjunto con el modelo LEL producido por el prototipo como documentación auxiliar, generando así una construcción híbrida que mejore el uso de recursos. Asimismo, debe tenerse presente que este prototipo no fue refinado después de su puesta a prueba en los dos casos de estudio, por lo que también es esperable mejorar los resultados de la herramienta basada en PLN, que es un trabajo a encarar a futuro.

Referencias

1. Antonelli, L., Rossi, G., Leite, J., Oliveros, A.: Deriving requirements specifications from the application domain language captured by Language Extended Lexicon. En: 15th Workshop on Requirements Engineering, Buenos Aires, Argentina (2012)
2. Mizuno, Y.: Software Quality Improvement. IEEE Computer, 16(3), 66-72 (1983)
3. Leite, J., Doorn, J., Kaplan, G., Hadad, G., Ridao, M.: Defining System Context using Scenarios. En: Perspectives on Software Requirements, pp.169-199. Kluwer Academic Publishers, Norwell, EE.UU. (2004)
4. Elizalde, M., Hadad, G., Doorn, J.: Incorporación de heurísticas lingüístico-cognitivas en el Proceso de Requisitos. En: 9° Congreso Nacional de Ingeniería Informática / Sistemas de Información, pp. 312-320. Mendoza, Argentina (2021)
5. Hadad, G., Doorn, J., Elizalde, M., García Ravlic, I., Casafuz, D.: Atender aspectos lingüístico-cognitivos en la captura de términos del contexto. En: XXIII Workshop de Investigadores en Ciencias de la Computación, pp. 410-414. Chilecito, Argentina (2021)
6. Hadad, G., Doorn, J., Elizalde, M., Ridao, M., Casafuz, D., Sebastián, A., Riera, G.: Impacto del

Proceso de las Entrevistas en la Calidad de los Modelos. En: 10° Congreso Nacional de Ing. Informática / Sistemas de Información, pp. 806-813. Concepción del Uruguay, Argentina (2022)
7. Litvak, C., Hadad, G., Doorn, J.: Procesamiento de Lenguaje Natural para Estudiar Completitud de Requisitos. En: XVIII Workshop de Investigadores en Ciencias de la Computación, pp. 498-502. Concordia, Argentina (2016)
8. Escudero, J.: Aplicación de la Inteligencia Artificial en la Ingeniería de Requisitos. Trabajo Final de Especialización. Universidad Católica Argentina, Buenos Aires, Argentina (2020)
9. Chowdhary, K.: Fundamentals of Artificial Intelligence. Springer New Delhi, India (2020)
10. Chopra, A., Prashar, A., Sain, C.: Natural Language Processing. International Journal of Technology Enhancements and Emerging Engineering Research, 1(4), 131-134 (2013)
11. Natural Language Toolkit, https://www.nltk.org/, último acceso 26/7/2023.
12. Two minutes NLP, https://medium.com/nlplanet/two-minutes-nlp-keyword-and-keyphrase-extraction-with-pke-5a0260e75f3e, último acceso 19/5/2023
13. spaCy, https://spacy.io/, último acceso 26/7/2023.
14. Google Colab, https://colab.google/ último acceso 24/7/2023
15. Mullen, L., Benoit, K., Keyes, O., Selivanov, D., Arnold, J.: Fast, Consistent Tokenization of Natural Language Text. Journal of Open Source Software, 3(23), 655 (2018)
16. Balakrishnan, V., Ethel, L.: Stemming and Lemmatization: A Comparison of Retrieval Performances. En: Lecture Notes on Software Engineering, pp. 262-267 (2014)
17. Wilbur J., Sirotkin, K.: The automatic identification of stop words. Journal of Information Science, 18(1), 45-55 (1992)
18. Li, W., Zhao, J.: TextRank algorithm by exploiting Wikipedia for short text keywords extraction. En: 3rd International Conference on Information Science and Control Engineering, pp. 683-686. IEEE (2016)
19. Mihalcea R., Tarau P.: TextRank: Bringing Order into Texts. En: Conference on Empirical Methods in Natural Language Processing, pp. 404-411 (2004)
20. Srinivasa-Desikan, B.: Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy and Keras. Packt Publishing (2018)
21. Vakare, T., Verma, K., Jain, V.: Sentence Semantic Similarity Using Dependency Parsing. En: 10th International Conference on Computing, Communication and Networking Technologies, pp. 1-4, Kanpur, India (2019)
22. Marneffe, M., Dozat, T., Silveira, N., Haverinen, K., Ginter F., Nivre, J., Manning, C.: Universal Stanford Dependencies: A cross-linguistic typology. En: International Conference on Language Resources and Evaluation, 14, 4585-4592 (2014)
23. Li, J., Sun, A., Han, J., Li.,C.: A Survey on Deep Learning for Named Entity Recognition. En: IEEE Transactions on Knowledge and Data Engineering, 34(1), 50-70. IEEE (2022)
24. Vidal Monge Navarro, N.M.: Ingeniería de Requerimientos: modelo de procesamiento de lenguaje natural para construcción del Léxico Extendido del Lenguaje, Trabajo Final de Carrera, Universidad de Belgrano, Buenos Aires, Argentina (2023)
25. Hadad, G., Doorn, J., Kaplan, G.: Creating Software System Context Glossaries. En: Encyclopedia of Information Science and Technology, II, 789-794. IGI Global, Mehdi Khosrow-Pour (ed), 2da. edición, Hershey, EE.UU. (2008)