ISBN 978 -987-633-574-4

Construcción de grafos de conocimiento a partir de especificaciones de requerimientos usando procesamiento de lenguaje natural

Luciana Tanevitch[1], Felipe Dioguardi[1], Juliana Delle Ville[1], Sebastián Villena[1], Francisco Herrera[1], Waldo Hasperue[2,4], Diego Torres[1,2,3], and Leandro Antonelli[1]

[1] LIFIA, Facultad de Informática, UNLP
[2] Comisión de Investigaciones Científicas (CICPBA)
[3] Departamento de Ciencia y Tecnología, UNQ
[4] Instituto de Investigación en Informática LIDI, Facultad de Informática, UNLP
luciana.tanevitch@lifia.info.unlp.edu.ar

Abstract. Los requerimientos cumplen un rol fundamental en el desarrollo de los sistemas informáticos, ya que un software bien diseñado y codificado no aporta ningún valor si no satisface los requerimientos. Relevar y especificar requerimientos no es una tarea fácil. La principal fuente de información son las personas y las técnicas basadas en lenguaje natural. Sin embargo, el lenguaje natural presenta muchas debilidades, por lo cual lograr una especificación de calidad es un gran desafío y esfuerzo. Así pues, se necesitan brindar herramientas para poder sintetizar las especificaciones de forma de poder identificar por ejemplo omisiones, ambigüedades y conflictos. Este artículo presenta una propuesta para construir un grafo de conocimiento a partir de una especificación en lenguaje natural. Se presenta tanto un proceso como una herramienta que automatiza el proceso. De esta forma, el grafo de conocimiento obtenido ofrece al analista una síntesis de los conceptos y las relaciones entre ellos.

Keywords: Grafo de conocimiento, Especificaciones de requerimientos, Lenguaje natural

1 Introducción

Un requerimiento [4] es la condición o capacidad que debe poseer un sistema o uno de sus componentes para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto. En otras palabras, un requerimiento describe las necesidades, deseos y expectativas [12] de los *stakeholders* (por ejemplo usuarios, o expertos del dominio) con el objetivo de lograr el producto adecuado para ellos [15]. De esta forma, los requerimientos cumplen un rol fundamental en el desarrollo de software, ya que los errores de la especificación de requerimientos se van a trasladar en los siguientes productos del ciclo de vida de desarrollo como por ejemplo diseño, código y casos de prueba. Un error en los requerimientos detectado cuando el software es entregado al usuario

ISBN 978 -987-633-574-4

implica volver a trabajar no solo de los requerimientos, sino también del diseño, codificación y testeo. Es así que esforzarse en lograr una buena especificación de requerimientos ayuda a reducir el volver a trabajar posteriormente. Es por eso que se recomienda que la especificación cumpla ciertos atributos de calidad como los siguientes: completa, concisa, coherente, consistente, no ambiguo y verificable entre otras [4].

De acuerdo a Loucopoulos et al. [12] la especificación de requerimientos es el producto final de un proceso con varias etapas: elicitación, modelización y validación. La elicitación es la etapa en la cual se obtiene el conocimiento necesario. Existen diferentes técnicas, sin embargo, lo más común es obtener el conocimiento de las personas. Esto se puede realizar a través de personas directamente utilizando entrevistas individuales o grupales, como así también en forma indirecta a través de cuestionarios o encuestas [15]. Para llevar a cabo este relevamiento es necesario que el equipo de desarrollo (los analistas) puedan comunicarse con los *stakeholders*. El medio de comunicación más utilizado es el lenguaje natural [3] ya que evita que los *stakeholders* aprendan formalismos que no les resultan naturales. De todas formas, el lenguaje natural presenta debilidades intrínsecas como ambigüedad, redundancia, incompletitud e inconsistencia, las que pueden ocasionar malos entendidos, que a su vez pueden ocasionar errores de especificación. Sea por ejemplo la frase "el pez esta listo para comer", la misma admite dos interpretaciones. Por un lado, se puede interpretar que el pez está listo para ser comido. Y por otro lado, se puede interpretar que el pez está listo para recibir su alimento. La diferencia entre ambas interpretaciones radica en si el pez es el sujeto que realiza la acción de comer, o en cambio una persona es quien come al pez (la persona es el sujeto mientras que el pez es el objeto).

Un grafo de conocimiento es un grafo que posee como propósito acumular y transmitir conocimiento sobre el mundo real, en el cual sus nodos representan entidades de interés y sus aristas relaciones entre esas entidades[10]. Para el ejemplo anterior, tanto "pez" como "persona" podrían ser nodos del grafo, mientras "comer" sería la relación entre ambos. De esta forma, quedaría muy claro que "la persona come al pez" y esto seria muy diferente de "el pez come un alga". Además de expresar el conocimiento de una forma más ordenada, los grafos de conocimiento le otorgan una semántica a cada nodo y cada relación. Es así que el significado sería preciso sin ocasionar ambigüedades ya que esta representación de conocimiento se basa en ontologías, que son formalizaciones que definen de forma estricta y no ambigua los conceptos que pretenden manejar [17]. Finalmente, un grafo de conocimiento brinda una especificación cuya interpretación puede automatizarse, es decir, se podrían utilizar algoritmos para inferir consecuencias a partir del grafo. Mas aún, los grafos de conocimiento permiten la implementación de técnicas para la resolución de sinónimos, búsquedas multi-idioma y resolución de ambigüedades. El objetivo de este artículo es presentar un proceso para procesar especificaciones en lenguaje natural y obtener un grafo de conocimiento. El artículo presenta una herramienta que da soporte al proceso.

ISBN 978 -987-633-574-4

El resto del artículo se organiza de la siguiente manera. La sección 2 analiza trabajos relacionados. La sección 3 describe grafos de conocimiento. La sección 4 presenta el proceso propuesto. Finalmente, la sección 5 discute conclusiones y trabajos futuros.

2 Trabajos relacionados

Delugach et al. [5] proponen un proceso para construir un grafo de conocimiento, sin embargo, el proceso es completamente manual y se basa en la interacción con los stakeholders. De todas formas, existen varios trabajos que proponen procesos automáticos o semi automáticos similares al nuestro. Yang et al. [23] proponen una herramienta para construir grafos de conocimiento para el lenguaje chino. Song et al. [19] también trabajan con el lenguaje chino y dadas las particularidades del mismo, ellos proponen una estrategia para construir mapas de conocimiento específicos de dominio a través de técnicas de deep learning. Qin et al. [16], quienes también trabajan con el lenguaje chino, proponen un método para construir un grafo de conocimiento a partir del lenguaje natural, pero específicamente para predecir enfermedades crónicas. Schlutter et al. [18] trabajan con oraciones simples para enriquecer un grafo de conocimiento y trabajan solo con el idioma inglés. Por su parte, Verma et al. proponen una técnica similar, sin embargo, ellos construyen mapas de conocimiento [22] o incluso generan ontologías luego de revisar las especificaciones y sugerir mejoras de redacción [21]. Otros trabajos utilizan los grafos de conocimiento como una herramienta para lograr otro fin. Hassan et al. [8] proponen una técnica para analizar requerimientos a partir de la construcción de grafos de requerimientos y finalmente lo convierten a un ontología. Ferrari et al. [6] sostienen la importancia del contexto para la detección de ambigüedades y para ello utilizan un grafo de conocimiento para representar el dominio y poder analizarlo. Mills et al. [13] además de utilizar lenguaje natura y grafos de conocimiento, ellos utilizan redes de Petri para enriquecer al análisis. Por otro lado, ellos convierten las oraciones en lenguaje natural, a una estructura "Agente, Acción, Paciente" lo cual es similar a nuestra estructura.

3 Grafos de conocimiento

En la actualidad, es necesario contar con estructuras que admitan el manejo de grandes volúmenes de datos, que puedan enriquecerse fácilmente, y que permitan, a través de su análisis, descubrir e inferir conocimiento implícito, más allá de las representaciones explicitadas en el esquema. Una manera de abordar esos requisitos es usando grafos de conocimiento [14]. Se puede visualizar a un grafo de conocimiento como una estructura de nodos que representan entidades, conectados a través de aristas que son las relaciones. Los grafos de conocimiento pueden, además, poser una definición del conocimiento con mayor nivel de formalismo a través de ontologías[10, 7]. Las ontologías son representaciones formales del conocimiento a través de taxonomías, que conceptualizan un dominio

ISBN 978 -987-633-574-4

determinado [9]. Su uso permite descubrir qué es una entidad, cómo deberían ser categorizadas, qué propiedades deberían tener, además de permitir realizar inferencias tales como "un perro es un animal, un animal es un ser vivo, los seres vivos tienen un tipo de alimentación, y por lo tanto un perro debe tener un tipo de alimentación". Para este tipo de descubrimiento, se requiere definir un conjunto de reglas a aplicar. Al introducir estos modelos semánticos, podemos ver que la semántica otorga significado a los datos, permite generar un contexto, permite manejar ambigüedades, lo que a su vez hace que la información en grafos de conocimiento pueda aprovecharse mejor, enriquecerse a lo largo del tiempo, y en consecuencia mejorar las consultas y análisis aplicables a ellos.

Al describir requerimientos necesitamos representar objetos del mundo real, no simples cadenas de texto que nombren un concepto, y para esto las propiedades de datos y de objetos ayudan a enriquecer la representación de un concepto. Podemos denotar a un grafo de conocimiento como un dataset de tripletas no ordenadas con la forma **entidad – relación – entidad**. Por ejemplo, para relaciones de elementos correspondientes a la administración de una provincia, podría existir la tripleta "La Plata – ubicada en – Buenos Aires", que determina que el nodo "La Plata" se relaciona con el nodo "Buenos Aires" a través de la relación "ubicada en". Gráficamente, las relaciones se pueden ver en un grafo (lo que le da el nombre a la estructura) como se muestra en la figura 1.

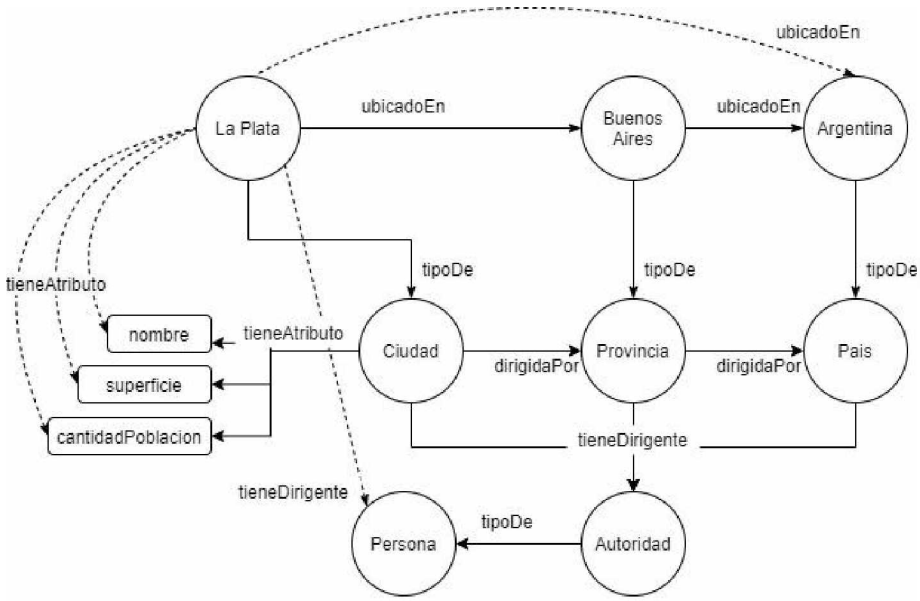

Fig. 1. Ejemplo grafo de conocimiento

Se pueden obtener ciertas conclusiones en base al modelo, tales como:

ISBN 978 -987-633-574-4

- La ciudad de La Plata está ubicada en la provincia de Buenos Aires.
- La provincia Buenos Aires está ubicada en el país Argentina.

Por lo tanto, si "ubicadoEn" es una propiedad transitiva, podemos inferir que la ciudad de La Plata está ubicada en el país Argentina.

- Una ciudad tiene una persona que es su autoridad.
- La Plata es una ciudad.

Entonces, la ciudad de La Plata tiene una persona que es su autoridad.

4 Proceso propuesto

Se plantea un proceso que recibe como entrada una especificación en lenguaje natural y produce como salida un grafo de conocimiento que sintetiza su información. Se propone la utilización de *kernel sentences* [11], frases simples creadas a partir de oraciones escritas en lenguaje natural, que mantienen la semántica original, y que siguen ciertas reglas sintácticas predefinidas. Determinar estas pautas no es un desafío menor, se deben producir expresiones fácilmente analizables por máquinas sin perder información relevante en el proceso. El proceso que se plantea en este trabajo consta de 4 etapas. En la primer etapa se separa el texto en oraciones. En la segunda etapa se reemplazan ciertas expresiones (pronombres personales, adjetivos demostrativos, etc.). En la tercera etapa se construyen kernel sentences que respetan una estructura simple del estilo "sujeto - verbo - objeto". La cuarta etapa identifica conceptos, propiedades y relaciones que usan para construir el grafo de conocimiento. La Figura 2 resume el proceso.

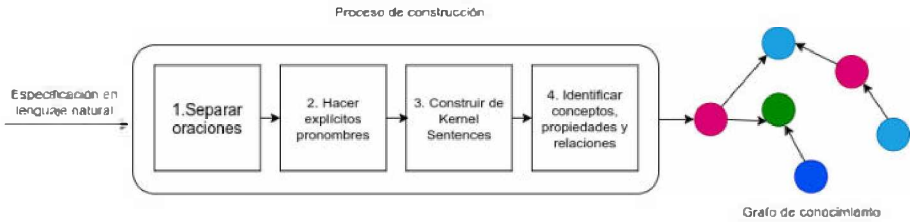

Fig. 2. Proceso de transformación

El resto de la sección amplía cada una de las etapas enumeradas. Para ello se utiliza la siguiente especificación.

Caso de estudio: *Sea una empresa que ofrece travesías en kayak. La misma ofrece distintas travesías que tienen diferente duración e itinerario. La empresa admite kayakistas inexpertos como también de experiencia. Cualquier persona le puede solicitar travesías a la empresa y la misma le informa el arancel.*

ISBN 978 -987-633-574-4

4.1 Separación de oraciones

El primer paso consiste en aplicar reglas que permiten delimitar dónde comienza una oración y dónde termina. La forma trivial es separarlas por signos de puntuación, por lo que, notando que la especificación contiene cuatro puntos, un primer procesamiento daría como resultado:

1. Sea una empresa que ofrece travesías en kayak.
2. La misma ofrece distintas travesías que tienen diferente duración e itinerario.
3. La empresa admite kayakistas inexpertos como también de experiencia.
4. Cualquier persona le puede solicitar travesías a la empresa y la misma le informa el arancel.

En algunas circunstancias, el punto puede ser utilizado para abreviaturas (*Sr.*, *Ud.*). Al igual que el signo de exclamación puede ser usado para interjecciones, y no para marcar una oración con las características que se analizaron (*¡Ay!*, *¡Viva!*). Se cuenta con herramientas de software que permiten reconocer estas situaciones, y así reducir la posibilidad de obtener expresiones que no cumplen con la estructura de una oración.

4.2 Explicitación de pronombres

A menudo, el lenguaje implica omisiones para evitar la repetición, el sujeto tácito es un ejemplo de esto. El sujeto tácito suele referir al núcleo de la expresión anterior, por lo que podría establecerse como regla de reemplazo de modo que el texto previo se reescriba de la siguiente manera:

1. Sea una empresa que ofrece travesías en kayak.
2. La empresa ofrece distintas travesías.
3. Las travesías tienen diferente duración e itinerario.
4. La empresa admite kayakistas inexpertos como también de experiencia.
5. Cualquier persona le puede solicitar travesías a la empresa y la empresa informa el arancel a la persona.

4.3 Construcción de kernel sentences

Las conjunciones agregan a las expresiones complejidad no deseada, por lo que se propone separar una oración que posea una conjunción en dos oraciones distintas. Se tuvieron en cuenta dos posibles escenarios: que la conjunción aparezca en el sujeto, o que aparezca en el objeto. En cualquiera de los casos se deben analizar las dependencias entre palabras para evitar la pérdida de información relevante al hacer el desglose. Por ejemplo, si hubiera modificadores del núcleo del objeto o del sujeto, podría nombrarse tácitamente luego de la conjunción, y si este fuera el caso debe tenerse en cuenta conservar el núcleo en las oraciones que se generen.

La especificación analizada tomaría la siguiente forma:

1. Sea una empresa que ofrece travesías en kayak.

ISBN 978 -987-633-574-4

2. La empresa ofrece distintas travesías.
3. Las travesías tienen diferente duración.
4. Las travesías tienen diferente itinerario.
5. La empresa admite kayakistas inexpertos
6. La empresa admite kayakistas de experiencia.
7. La persona puede solicitar travesías a la empresa.
8. La empresa informa el arancel a la persona.

En la frase "La empresa admite kayakistas inexpertos como también de experiencia.", la palabra *kayakista*, el núcleo del objeto, debe aparecer en las oraciones extraídas. De no ser así, podría ser complejo inferir qué es lo que la empresa admite "de experiencia".

La oración "Cualquier persona le puede solicitar travesías a la empresa y la empresa informa el arancel a la persona.", se puede separar en la conjunción y obtener dos sentencias que cumplen el requisito "sujeto - verbo - objeto".

4.4 Identificación de conceptos, propiedades y relaciones

Para lograr identificar las relaciones de dependencia se utilizan herramientas de procesamiento de lenguaje natural que permiten asignar etiquetas a cada una de las palabras para que las máquinas puedan reconocerlas y analizarlas. Partiendo de una estructura sintáctica del tipo "sujeto - verbo - objeto" se pueden aplicar nuevas reglas que permitan reconocer entidades, y las relaciones entre ellas. Entonces, podemos ver al núcleo del sujeto como el ejecutor de una acción, quien será identificado como una entidad. El objeto directo será reconocido como una entidad en caso de que el objeto indirecto no esté presente. Caso contrario el objeto indirecto, el receptor de una acción, será una entidad. El verbo "tener" indica una relación de posesión o pertenencia por parte de la entidad que refiere el sujeto; este posibilita reconocer como propiedades de una entidad aquellas expresiones involucradas con el verbo (regularmente, el objeto directo). Las relaciones pueden ser un verbo, una frase verbal, o la combinación del verbo con el objeto directo (en caso que esté presente el objeto indirecto). Para permitir comprender mejor este proceso, se describirá en la siguiente sección una herramienta que lo lleva a cabo.

5 Herramienta implementada

Con el fin de dar soporte automatizado al proceso propuesto, se implementó una herramienta Web en Python que utiliza la librería Spacy [20] como soporte al procesamiento de lenguaje natural[5]. En la figura 3 se puede ver una captura de la página principal de la herramienta. A continuación se describe el funcionamiento de esta herramienta.

[5] El código se encuentra disponible en `https://github.com/cientopolis/requirements-knowledge-graph`

ISBN 978 -987-633-574-4

Fig. 3. Herramienta, home page

En primera instancia, toma el texto ingresado y utiliza Spacy para obtener el rol sintáctico de cada palabra. A continuación, se aplican las reglas de transformación propuestas en la sección 4 para obtener las kernel sentences equivalentes a las oraciones originales.

A partir de estas expresiones, la herramienta identifica entidades, propiedades y relaciones siguiendo lo definido en la sección 4.4, y construye un grafo de conocimiento que almacena en formato RDF [1]. Luego permite realizar gráficos representativos y consultas mediante el lenguaje SPARQL [2].

Para comprender cómo trabaja la herramienta, se analizará su funcionamiento dadas siguientes kernel sentences:

1. La empresa ofrece travesías en kayak.
2. Las travesías en kayak tienen arancel.
3. Los kayakistas contratan travesías en kayak.
4. La empresa informa el arancel a los kayakistas.

Oración 1: se identifica *empresa* como entidad por ser el núcleo del sujeto, y *travesías en kayak* por objeto directo, en ausencia del objeto indirecto. La empresa realiza la acción de *ofrecer*, que sirve como vínculo con la entidad *travesías en kayak*. Es interesante notar que se selecciona *travesías en kayak*, siendo *en kayak* un modificador del sustantivo, porque agrega precisión y contexto. Ignorarlo produciría un resultado demasiado genérico, que es lo que se busca evitar.

Oración 2: se identifica la entidad *travesías en kayak*, y su atributo *arancel*, siguiendo las reglas propuestas en el proceso que indican reconocer como propiedades aquellas expresiones que se asocien al verbo "tener".

Oración 3: se identifica la entidad *kayakistas*, que realiza la acción *contratar* que la vincula con la entidad *travesías en kayak*.

Oración 4: se identifica la entidad *empresa* por ser núcleo del sujeto, y como hay presencia de objeto directo, también lo será *kayakistas*, el núcleo del objeto indirecto. Entonces, la relación será *informa arancel*, siendo la combinación del verbo con el objeto directo.

La figura 4 muestra el resultado del análisis realizado. Allí puede verse que se detectan las entidades y las acciones que las vinculan. Además se detectan algunas características que pueden describirse en términos de programación orientada a objetos como es el caso de Empresa como una clase candidata.

ISBN 978 -987-633-574-4

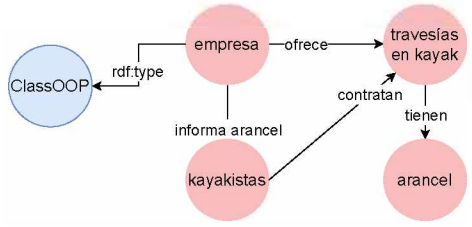

Fig. 4. Herramienta de soporte del proceso de transformación

6 Conclusiones

Este articulo presentó un proceso que tiene como finalidad la construcción de un grafo de conocimiento a partir de una especificación en lenguaje natural. El proceso cuenta con una herramienta que le da soporte a cada una de las cuatro etapas que lo conforman. El proceso propuesto permite realizar una síntesis de las conceptos y las relaciones a través de un grafo, de forma que el analista pueda identificar los aspectos que necesitan ser mejorados en la especificación. La herramienta presentada tiene como finalidad el tratamiento del lenguaje español y se basa en una librería de terceros, pero tuvo que ser enriquecida para ciertas particularidades del lenguaje español, que dada lo rico que es y las distintas variantes para expresar las ideas, lo convierten en un lenguaje muy complejo de analizar. Nuestros próximos trabajos tienen varias arista. Por un lado mejorar los aspectos de procesamiento del lenguaje español. Por otro lado, queremos vincular el grafo de conocimiento producido por nuestro enfoque con otros grafos de conocimientos ya elaborados, de forma de poder aprovechar el conocimiento en los otros grafos. Finalmente, estamos trabajando en analizar el grafo de conocimiento, para identificar automáticamente debilidades y poder sugerir al analista que aspectos de la especificación necesitan ser mejorados. También se está trabajando en detectar a partir de las especificaciones más elementos del paradigma orientada a objetos como posibles métodos y atributos.

References

1. RDF. `https://www.w3.org/RDF/`, accedido: 2020-08-01
2. SPARQL. `https://www.w3.org/TR/sparql11-query/`, accedido: 2020-08-01
3. Alzayed, A., Al-Hunaiyyan, A.: A bird's eye view of natural language processing and requirements engineering
4. Committee, I.C.S.S.E.S., Board, I.S.S.: Ieee recommended practice for software requirements specifications, vol. 830. IEEE (1998)
5. Delugach, H., Lampkin, B.: Acquiring software requirements as conceptual graphs. In: Proceedings Fifth IEEE International Symposium on Requirements Engineering. pp. 296–297 (2001)
6. Ferrari, A., Gnesi, S.: Using collective intelligence to detect pragmatic ambiguities. In: 2012 20th IEEE International Requirements Engineering Conference (RE). pp. 191–200. IEEE (2012)

ISBN 978 -987-633-574-4

7. Guarino, N., Oberle, D., Staab, S.: What Is an Ontology?, pp. 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-540-92673-3_0
8. Hassan, T., Hassan, S., Yar, M.A., Younas, W.: Semantic analysis of natural language software requirement. In: 2016 Sixth International Conference on Innovative Computing Technology (INTECH). pp. 459–463 (2016)
9. Hogan, A.: The Web of Data. Springer (2020), https://doi.org/10.1007/978-3-030-51580-5
10. Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge graphs. ACM Comput. Surv. 54(4) (Jul 2021), https://doi.org/10.1145/3447772
11. Katz, B.: From sentence processing to information access on the world wide web
12. Loucopoulos, P., Karakostas, V.: System Requirements Engineering. McGraw-Hill, Inc., USA (1995)
13. Mills, M., Psarologou, A., Bourbakis, N.: Modeling natural language sentences into spn graphs. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence. pp. 889–896 (2013)
14. Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: Lessons and challenges. Commun. ACM 62(8), 36–43 (Jul 2019), https://doi.org/10.1145/3331166
15. Pohl, K.: Requirements Engineering Fundamentals, 2nd Edition: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant. Rocky Nook-Ips (2016), https://books.google.com.ar/books?id=1VsUDgAAQBAJ
16. Qin, S., Xu, C., Zhang, F., Jiang, T., Ge, W., Li, J.: Research on application of chinese natural language processing in constructing knowledge graph of chronic diseases. In: 2021 International Conference on Communications, Information System and Computer Engineering (CISCE). pp. 271–274 (2021)
17. Saorín, T.: Grafos de conocimiento y bases de datos en grafo: conceptos fundamentales a partir de una" obra maestra" del museo del prado. Anuario Think EPI 13 (2019)
18. Schlutter, A., Vogelsang, A.: Knowledge representation of requirements documents using natural language processing (2018)
19. Song, Y., Rao, R.N., Shi, J.: Relation classification in knowledge graph based on natural language text. In: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). pp. 1104–1107 (2018)
20. Vasiliev, Y.: Natural Language Processing with Python and SpaCy: A Practical Introduction. No Starch Press (2020)
21. Verma, K., Kass, A.: Requirements analysis tool: A tool for automatically analyzing software requirements documents. In: International semantic web conference. pp. 751–763. Springer (2008)
22. Verma, R.P., Beg, M.R.: Representation of knowledge from software requirements expressed in natural language. In: 2013 6th International Conference on Emerging Trends in Engineering and Technology. pp. 154–158 (2013)
23. Yang, X., Zhao, S., Cheng, B., Wang, X., Ao, J., Li, Z., Cao, Z.: A general solution and practice for automatically constructing domain knowledge graph. In: 2020 IEEE 6th International Conference on Computer and Communications (ICCC). pp. 1675–1681 (2020)