EasyChair Preprint

№ 9671

NLP Tool for the Detection of Ambiguities in Software Requirements Written in Spanish

Samira Enriquez, Jonathan Ramírez Reyes, Dunia Colomé Cedeño, Reiman Alfonso Azcuy and Héctor González Diez

April 27, 2023

//UCIENCIA/23
CONVENCIÓN CIENTÍFICA INTERNACIONAL

Herramienta de PNL para la detección de ambigüedades en requisitos de software escritos en español

NLP tool for the detection of ambiguities in software requirements written in Spanish

Samira Enríquez González [1], *Jonathan Ramírez Reyes* [2], *Dunia Colomé Cedeño* [3*], *Reiman Alfonso Azcuy* [4], *Héctor González Diez* [5]

[1] *Universidad de Ciencias Informáticas*. samiradlmeg@estudiante.uci.cu
[2] *Universidad de Ciencias Informáticas*. *jonathanrr@estudiante.uci.cu*
[3] *Universidad de Ciencias Informáticas*. *dcolome@uci.cu*
[4] *Universidad de Ciencias Informáticas*. *razcuy@uci.cu*
[5] *Universidad de Ciencias Informáticas*. *hglez@uci.cu*

[*] Autor para correspondencia: *dcolome@uci.cu*

Resumen

La ingeniería de requisitos es una de las etapas más importantes del ciclo de vida del desarrollo de software. El éxito de cualquier producto de software depende de la calidad de sus requisitos. Los requisitos de software suelen estar escritos en lenguaje natural. La ambigüedad en los requisitos escritos en lenguaje natural es un problema que ha sido estudiado por la comunidad de ingeniería de requisitos durante más de dos décadas. La resolución manual de la ambigüedad en los requisitos es tediosa y requiere mucho tiempo. Existen varias herramientas de procesamiento del lenguaje natural para automatizar el análisis de la ambigüedad, sin embargo, la mayoría de ellas no están ampliamente disponibles, son obsoletas, poco seguras y caras; las pocas herramientas públicas sólo permiten el análisis de requisitos en lengua inglesa. Esta investigación pretende desarrollar una herramienta de procesamiento del lenguaje natural para detectar la ambigüedad léxica y sintáctica presente en los requisitos de software, utilizando el lenguaje de programación Python y herramientas de procesamiento del lenguaje natural como NLTK. Como resultado de este trabajo, se presenta un conjunto de datos que contiene 19.357 requisitos pertenecientes a los proyectos de desarrollo de software de la Universidad de Ciencias Informáticas; el conjunto de datos obtenidos constituye una línea de base para futuras investigaciones. Se utilizó la metodología XP para guiar el desarrollo de la herramienta propuesta. Se evaluó el

enfoque en un conjunto de datos de 100 requisitos y logramos un 98% de precisión y un 91% de exhaustividad.
Palabras clave: ambigüedad, dataset procesamiento del lenguaje natural, requisitos de software, técnicas.

Abstract

Requirements engineering is one of the most important stages of the software development life cycle. The success of any software product depends on the quality of its requirements. Software requirements are usually written in natural language. Ambiguity in requirements written in natural language is a problem that has been studied by the requirements engineering community for more than two decades. Manual resolution of ambiguity in requirements is tedious and time consuming. Several natural language processing tools exist to automate ambiguity analysis, however, most of them are not widely available, obsolete, unsafe, and expensive; the few public tools only support English language requirements analysis. This research aims to develop a natural language processing tool to detect lexical and syntactic ambiguity present in software requirements, using the Python programming language and natural language processing tools such as NLTK. As a result of this work, a dataset containing 19,357 requirements belonging to software development projects at the University of Computer Science is presented; the obtained dataset constitutes a baseline for future research. The XP methodology was used to guide the development of the proposed tool. The approach was evaluated on a data set of 100 requirements and we achieved 98% accuracy and 91% completeness.
Keywords: *ambiguity, dataset, natural language processing, software requirements, techniques.*

Introducción

La especificación de los requisitos del software debe definirlos de forma precisa, completa y verificable. Los requisitos del software se especifican generalmente en lenguaje natural, por lo que asumen los problemas que tiene este lenguaje, como la vaguedad y la ambigüedad. Los defectos en los requisitos de los programas informáticos son una causa fundamental de insatisfacción de clientes y usuarios debido a la entrega de productos acabados que no responden a sus necesidades e intereses. La evaluación de la calidad de los requisitos en lenguaje natural suele seguir un modelo específico, basado en determinados criterios de calidad.

Entre los criterios de calidad abordados por diversos autores para la evaluación de requisitos, se encuentran la **claridad** (Gnesi y Trentanni, 2019), (Naeem, et al. 2019), (Ferrari, et al. 2019), **corrección** (ISO/IEC /IEEE29148:2018),

(Ferrari, et al.2019), (Naeem, et al. 2019), (Kumar, et al. 2019), **no ambigüedad** (ISO/IEC/IEEE29148:2018), (Gnesi y Trentanni,2019), (Ferrari, et al. 2019), (Arendse y Lucasen, 2016), (Kumar, et al. 2019), **exhaustividad** (ISO/IEC/IEEE 29148:2018), (Kumar, et al. 2019), (Ferrari, et al. 2019),**singularidad** (ISO/IEC/IEEE 29148:2018), (Ferrari, et al. 2019) y otros como **atomicidad, factibilidad, verificable, apropiado, comprensibilidad y consistencia**. Además, la norma ISO/IEC/IEEE29148:2018 reconoce como criterios de calidad para un conjunto de requisitos los siguientes: completo, coherente, factible, comprensible y susceptible de ser validado. El criterio de no ambigüedad tiene un interés significativo en los requisitos de software, por lo que es el criterio que se aborda en esta investigación. Además, existen criterios lingüísticos utilizados por diferentes herramientas de PNL (procesamiento del lenguaje natural) para RI (ingeniería de requisito). Entre los criterios lingüísticos, destaca el uso de **superlativos** (ISO/IEC/IEEE 29148:2018), (Arendse y Lucasen, 2016), **términos subjetivos** (ISO/IEC/IEEE 29148:2018), (Gnesi y Trentanni, 2019),(Arendse y Lucasen, 2016), (Ferrari, et al. 2019), pronombres vagos (ISO/IEC/IEEE 29148:2018), (Gnesi y Trentanni, 2019), (Arendse y Lucasen, 2016), (Naeem, et al. 2019), **frases o términos comparativos** (ISO/IEC/IEEE 29148:2018), (Arendse y Lucasen, 2016), **imperativos** (Gnesi y Trentanni, 2019), (Arendse y Lucasen, 2016), (Naeem, et al. . 2019), **opcionales** (Gnesi y Trentanni, 2019), (Naeem, et al. 2019), (Ferrari, et al. 2019), **cuantificadores** (Arendse y Lucasen, 2016), (Naeem, et al. 2019), términos ambiguos (ISO/IEC/IEEE 29148:2018), (Arendse y Lucasen, 2016), **términos negativos** (Arendse y Lucasen, 2016), (Naeem, et al. 2019), entre otros.

Investigaciones anteriores *(Zhao et al. 2022) (Falessi, Cantone, y Canfora 2018) (Tjong y Berry, 2018)* han presentado sistemas que emplean técnicas de procesamiento del lenguaje natural (por ejemplo, Part-of-Speech, Lexical Patterns, Stemming, Lemmatization) y herramientas (por ejemplo, Standford CoreNLP, GATE, NLTK, Spacy) para resolver la tarea de detección de defectos en los requisitos de software. Sin embargo, la mayoría de estos sistemas no están ampliamente disponibles, son anticuados, poco seguros y caros; las pocas herramientas públicas sólo permiten el análisis de requisitos en lengua inglesa.

Por otro lado, los dos componentes fundamentales para el procesamiento del lenguaje natural son los datos y los algoritmos. En este sentido, la mayoría de los conjuntos de datos disponibles están en lengua inglesa (por ejemplo, Kaggle y PURE), de ahí la necesidad de crear un conjunto de datos con requisitos en lengua española.

Ambigüedad en los requisitos del software

La ambigüedad es el término que hace referencia a aquellas estructuras gramaticales que pueden entenderse de varias maneras o admiten diferentes interpretaciones y, por tanto, dan lugar a dudas, incertidumbre o confusión. *(Ortuño Sánchez 2016)* Los requisitos de software pueden tener diferentes tipos de ambigüedad Figura. 1. La ambigüedad de los requisitos de software puede ser de diferentes tipos, como la léxica, la sintáctica, la semántica y la pragmática. *(Osama et al. 2020)*

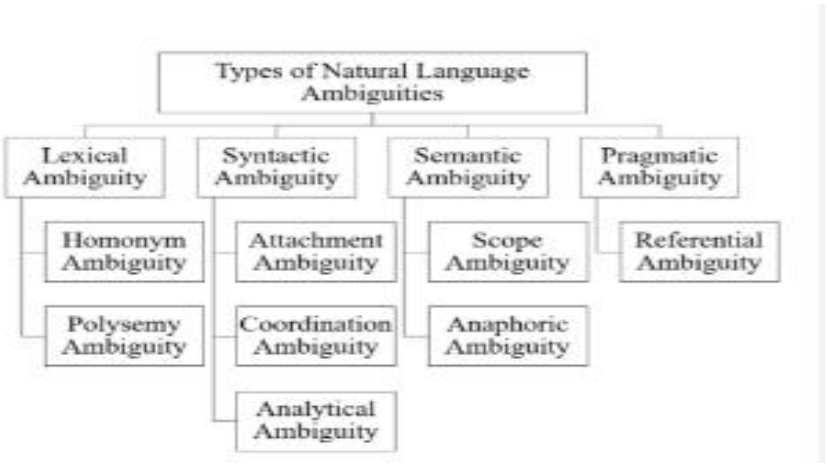

Figura 1 Tipos de ambigüedad (Osama et al. 2020)

En este trabajo se tratan las ambigüedades sintácticas y léxicas. A continuación, se detallan las ambigüedades léxicas y sintácticas.

A. Ambigüedad léxica

La ambigüedad léxica se refiere a las palabras o frases que tienden a tener más de un significado dependiendo del contexto y esto puede ser del tipo homónimo o polisémico. La homonimia se produce cuando varias palabras tienen la misma ortografía o fonética, pero significados diferentes. Por ejemplo, lo que sucede con llama-fuego y llama-animal, que derivan del latín. Por otro lado, la polisemia se produce cuando una misma palabra da diferentes significados en distintos contextos. Esto se evidencia con la palabra periódico puede significar 1) publicación física impresa “el periódico se mojó con la lluvia”, 2) la ilustración editora “el periódico firmó a su personal de edición”.(Zhao et al. 2022) (Osama et al. 2020)

B. Ambigüedad sintáctica

La ambigüedad sintáctica se manifiesta cuando una expresión o frase tiene más de una estructura gramatical. Existen tres tipos de ambigüedad: sintáctica coordinativa, de apego y analítica(Osama et al. 2020). La coordinativa (López 2019) se puede presentar cuando una oración contiene más de una palabra de tipo conjunción; puede ser copulativa, disyuntiva o mixta. La ambigüedad de apego ocurre cuando existe la duda de la unión de una frase con otra. Por último, la ambigüedad analítica, se produce cuando el papel de los constituyentes dentro de una oración o frase es ambiguo, esto es común en las oraciones sustantivas compuestas. (Osama et al 2020)

Para la detección y reducción de la ambigüedad en la documentación de requisitos, se identificaron distintas soluciones que pueden mejorar la calidad de la especificación de requisitos. Estas soluciones están agrupadas en las seis categorías siguientes (Hayman Oo et al. 2018)

- Placas de caldera
- Lenguajes controlados
  - Inspecciones
  - Orientada a objeto y basada en UML
  - Ontología
  - Procesamiento del lenguaje natural (PLN)

La mayor parte de las herramientas automatizadas se basan en soluciones de ontología y PLN. Las soluciones basadas en ontologías son una conceptualización de un dominio que permite la definición de relaciones semánticas entre entidades y la inferencia de conocimiento a través del razonamiento. Esta solución se utiliza para la detección de ambigüedades léxicas, además de ser adecuada para la elicitación de requisitos y puede ayudar a reducir el esfuerzo.*(Kumar et al. 2019)*

A partir del significado de las ambigüedades léxicas y sintácticas y de sus relaciones con varios de los criterios de lenguaje de requisitos presentados anteriormente, se ha definido el siguiente modelo de calidad para la evaluación de requisitos en lenguaje natural, que se ha utilizado en SIDARES.

Criterio de calidad: No hay ambigüedad.

Indicadores según el tipo de ambigüedad:

La ambigüedad léxica aparece cuando una palabra tiene diferentes significados.

- El uso de múltiples frases dará lugar a diversas interpretaciones.
- No debe haber términos o frases con múltiples interpretaciones.
- Palabras o frases con más de un significado que causan confusión.

La ambigüedad sintáctica aparece cuando una frase u oración está asociada a más de una estructura gramatical.

- No deben aparecer conjunciones disyuntivas (o, u, sea, bien) ni copulativos (y, e, o, que).
- No debe haber preposiciones separables (a, con, de, en).

Procesamiento del lenguaje natural para detectar defectos en los requisitos del software

El PLN es una tecnología que ha ido evolucionando en los últimos años. Su campo de acción se encuentra dentro de la Inteligencia artificial y la lingüística, para interpretar el LN en la relación de los seres humanos y las máquinas.

La relación entre el PLN y la ingeniería de requisitos está establecida y ampliamente discutida *(Zhao et al 2022)*. El PLN para la IR (PLN4IR) o el NLP4RE (por sus siglas en inglés) es un área de investigación y desarrollo que busca aplicar tecnologías de PLN (técnicas, herramientas y recursos) a diferentes tipos de documentos de requisitos para respaldar una variedad de tareas de análisis lingüístico realizadas en varias etapas de la IR. *(Guzmán-Luna et al. 2018)* En los últimos 20 años, más de 50 herramientas han sido desarrolladas, ya sean automatizadas o semiautomatizadas, que emplean técnicas de PLN para mejorar la calidad de los requisitos *(Naeem et.al 2019)*. Existen cuatro tipos de clasificación de los enfoques de las herramientas de PLN para la IR *(Soren 2020)*, los cuales se mencionan a continuación:

1. Resaltar los defectos y desviaciones en el documento de requisitos escrito en lenguaje natural.
2. Generación de modelos a partir de la descripción del lenguaje natural.
3. Determinar los vínculos de rastreo entre la descripción de lenguaje natural.
4. Encontrar las abstracciones clave de los documentos en lenguaje natural.

Muchas de estas herramientas están obsoletas, entre ellas SHREE, QUARS, SAT y AQUSA *(Naeem et.al. 2019)*, otras se encuentran activas. En la Tabla 1 se muestra un resumen de las herramientas de PLN para IR propuesto por Afrah Naeen, Zeeshan Aslam y Munam Ali Shah (Naeem et.al. 2019), el cual se ha tomado como base para la selección de las herramientas a analizar para desarrollar SIDARES.

Tabla 1Resumen de herramientas de PLN para IR. (Naeem et.al. 2019)

Nombre de la herramienta	Año	Disponibilidad

Qvscribe	2016	Licenciada
Innoslate	2014	Libre
QUOD	2014	Libre
RQA	2011	Licenciada
Requirement Assesment tool (RAT)	2018	Libre
TACTILE check	2017	Libre
Tiger Pro	2004	Libre para estudiantes
QUALICEN	2014	Licenciada
Visure quality analyzzer	2011	De pago
NASA reconstructed ARM	2011	No disponible
Pragmatic Ambiguity Detector	2012	Libre
IntelliReq	2014	Bloqueada
NARCIA	2015	Instalable en la PC
AQUSA	2016	Libre solo para su comunidad

Una tarea de PLN se define como "*un trabajo de procesamiento de texto que se puede realizar mediante una o más técnicas de PLN, con el apoyo de algunas herramientas y recursos de PLN*" *(Zhao et al. 2022)* .Mientras que, las tareas del PLN para IR son tareas de análisis lingüístico llevadas a cabo durante las actividades de la IR*(Guzmán-Luna et al. 2018).*

Dentro de las tareas de la PLN con un enfoque orientado a la IR se encuentran la **detección** *(Tjong et al. 2018) (Falessi et al. 2018),* **extracción** *(Zhao et al. 2022),* **clasificación** *(Casamayor, et al. 2018),* **modelado** *(Ambriola et al. 2018),* **rastreo y relación** *(Zhao et al. 2022),* **búsqueda y recuperación** (Morales-Ramirez, Kifetew, y Perini 2019) *(Zhao et al. 2022).*Cada una de estas tareas tienen un objetivo específico, que aporta facilidad a la hora de dar cumplimiento a distintas actividades de la IR. Para la herramienta propuesta se decidió realizar la tarea **detección**.

La detección ayuda a detectar problemas lingüísticos en los documentos de requisitos. Además, suele servir de apoyo a las actividades de revisión manual para que los requisitos, o los artefactos relacionados con ellos, sean claros e inequívocos. También se puede incluir en esta tarea la comprobación del cumplimiento de plantillas de requisitos predefinidas y la identificación de requisitos equivalentes, ya que el objetivo sigue siendo aplicar el rigor en los textos de requisitos.

Una técnica de PLN es "*un método práctico, un enfoque, un proceso o un procedimiento para realizar una tarea de PLN concreta"*. *(Zhaoetal.2021), (Zhaoetal.2022)*

A continuación, se describen en la Tabla 2, algunas de las técnicas de PLN más recientes, publicadas en los últimos dos años, utilizadas con frecuencia en la IR *(Zhao et al.2022).*

Tabla 2: Técnicas de PLN más empleadas en la IR.

Nombre	Explicación
Etiquetado de parte del discurso (clasificación)	El etiquetado de parte del discurso procesa una secuencia de palabras, y adjunta una etiqueta a cada palabra. Las partes de la oración también se conocen como clases de palabras o categorías léxicas.
Chunking	Es un tipo de análisis sintáctico superficial que analiza una frase, identificando primero sus partes constituyentes (sustantivos, verbos, adjetivos, etc.) y luego los relaciona con unidades de orden superior que tienen significados gramaticales discretos (grupos de sustantivos o frases, grupos de verbos, etc.).
Análisis del sentimiento	Es el proceso de identificar y categorizar computacionalmente las opiniones expresadas en un texto.

Indexación semántica latente	Una práctica matemática que ayuda a clasificar y recuperar información sobre determinados términos clave y conceptos utilizando la descomposición del valor singular.
Patrones semánticos	Los patrones semánticos se generan a partir de los conceptos comunes que coinciden. Los conceptos más coincidentes de cada palabra, son considerados. Un patrón semántico puede relacionarse con varios conceptos y una sola estructura semántica puede contener varios patrones semánticos.
Patrones léxicos	Palabras o fragmentos de texto que aparecen en el lenguaje con alta frecuencia y el significado de las partes del texto o la palabra son a veces diferentes del significado del texto o la palabra en su conjunto.
Detección de homónimos	Detectar las palabras que se pronuncian igual entre sí (por ejemplo, "caza" y "casa") o tienen la misma grafía (por ejemplo, "llama de fuego" y "llama de animal")
Análisis basado en reglas	Utiliza reglas gramaticales, reglas semánticas o patrones para analizar la sintaxis de un texto.
Normalización del texto	Convertir las palabras en su forma original y eliminar las palabras o caracteres innecesarios del texto.
Segmentación de textos	Descomponer un texto en una secuencia de frases o palabras individuales.
Lematización	Utilizar un análisis de vocabulario y morfológico de las palabras, normalmente con el objetivo de eliminar las inflexiones y devolver la forma base o de diccionario de una palabra, que se conoce como lema.
Eliminación del ruido	Eliminación de caracteres, dígitos y trozos de texto que puedan interferir en el análisis del texto.
Tokenización	El proceso de dividir una cadena de texto en palabras, frases, símbolos u otros tokens

	significativos.
Frecuencia de las palabras	La frecuencia con la que aparece una palabra en un documento, dividida por el número de palabras que tiene.

Teniendo en cuenta el análisis de las características y el uso de las técnicas anteriormente descritas para la IR. Se decide utilizar en la implementación de los algoritmos de detección de ambigüedades léxicas y sintácticas de la herramienta SIDARES, la **tokenización**, la **eliminación del ruido** y el **etiquetado de parte del discurso**. El uso de estas técnicas es muy común en el PLN para el preprocesamiento. La tokenización es la técnica más importante y obligatoria a la hora del tratamiento de texto (Ramírez 2018) (Ricardo Javier. et al. 2021), porque separa los fragmentos de texto en pequeñas unidades. Esto permite un procesamiento y limpiado del texto más eficiente. La eliminación de ruido permite la limpieza adecuada del texto y el objetivo del etiquetado de parte del discurso es asignar una clasificación a cada token como se describe en la tabla anterior. Una clasificación para un token puede ser por ejemplo sustantivo o verbo. Dentro de los indicadores de calidad sintácticos se encuentran la no existencia de conjunciones ni preposiciones por eso esta técnica sería de gran utilidad.

Enfoque de la herramienta

La herramienta informática que se propone permite identificar la existencia de términos ambiguos en requisitos de software. Los requisitos analizados por esta, pueden ser redactados de forma manual o importados desde un documento con extensión (.xlsx,.csv). La herramienta una vez que obtiene y analiza los requisitos, es capaz de mostrar el requisito con la ambigüedad que detectó ya sea léxica o sintáctica, con el propósito de que la palabra o frase ambigua pueda ser corregida por el ingeniero de requisitos. La figura 2 muestra la interfaz de la herramienta propuesta. Esta interfaz está compuesta por un conjunto de componentes visuales. En la parte superior izquierda se encuentra el nombre del sistema. SIDARES, cuenta con los botones “Procesar requisito” y “Procesar columna “, los cuales son los encargados de analizar el requisito y documento de requisitos para así detectar ambigüedades léxicas o sintácticas. El botón Cargar documento es el que permite importar el documento de requisito para ser posteriormente analizado. En la parte inferior del contenedor principal se encuentra una ventana que muestra los requisitos con las ambigüedades detectadas, además del botón “Eliminar lista de ambigüedades” para vaciar este campo si así lo desea el usuario.

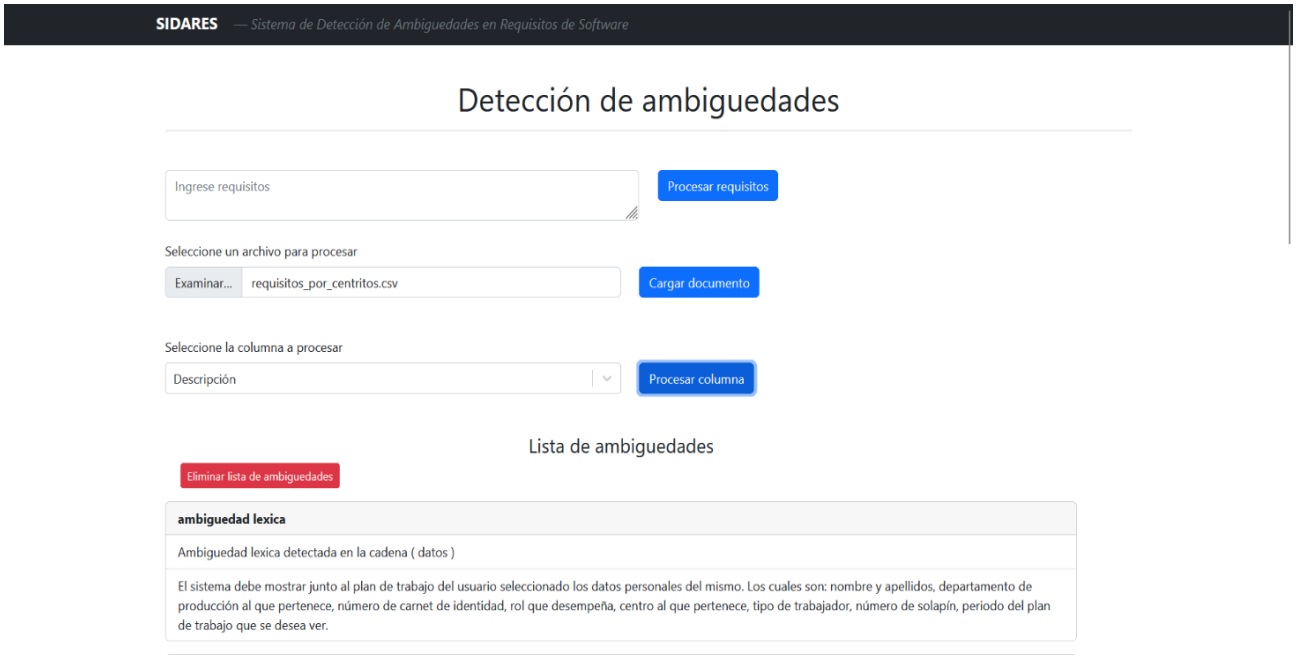

Figura 2 Interfaz de SIDARES

Flujo del proceso PLN

Este enfoque aplicó una línea de procesamiento del lenguaje natural compuesta por tres técnicas de PLN: 1-) tokenización para dividir el texto en tokens, 2-) eliminación de ruido como, caracteres no deseados o espacios en blanco y números, 3-) clasificación para asignar una etiqueta por parte del discurso (POS), por ejemplo, sustantivo, verbo o pronombre, a cada token de cada frase. La figura 3 muestra como ocurre el preprocesamiento haciendo uso de las técnicas anteriormente mencionadas. Como entrada se tiene al documento de requisitos o al requisito escrito manualmente. Luego se pasa a la fase de procesamiento del contenido textual donde se aplican las técnicas de PLN que se muestran en la figura 3. Una vez terminado el proceso de PLN se determina que el requisito de software presenta ambigüedad.

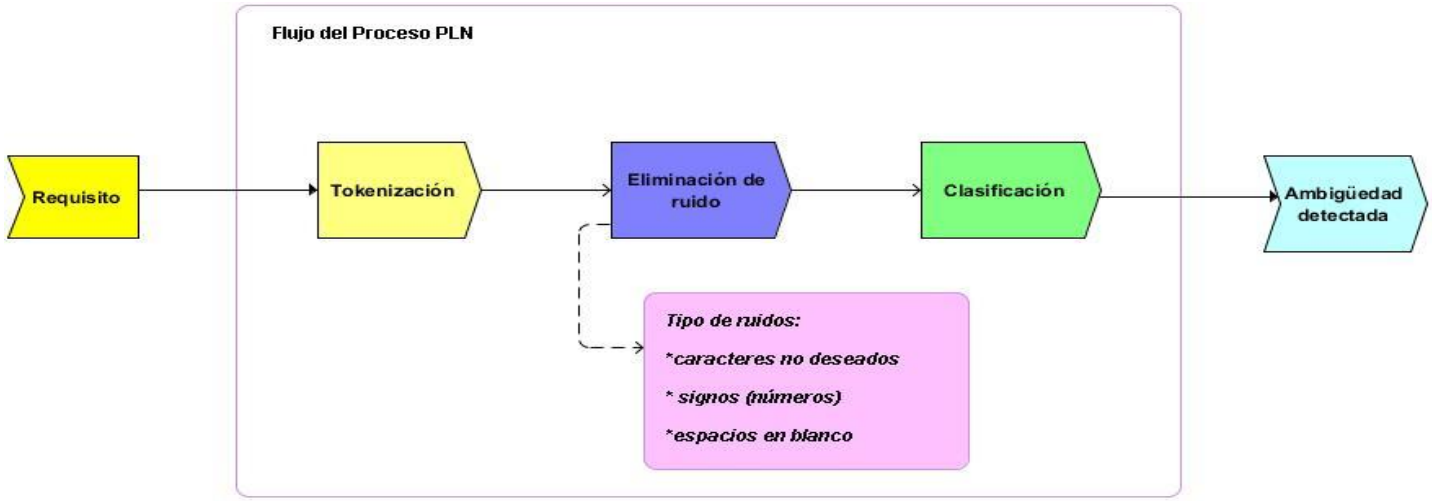

Figura 3 Flujo PLN

Para la detección de ambigüedad léxica y sintáctica en los requisitos se implementaron 2 algoritmos basados en reglas heurísticas y el modelo de calidad definido. A continuación, se definen las reglas empleadas en el enfoque propuesto.

Detección de ambigüedad léxica

Para comprobar la presencia de ambigüedad léxica en un requisito, se defino una regla. Sean A conjuntos de términos ambiguos porque tienen más de una interpretación, S un requisito, y sea T (S) cualquier secuencia de palabras del requisito. La regla es la siguiente:

R1: ∀a ∈ A, ∀t ∈ T (S), si t = a, marca t como ambigüedad léxica.

Si un requisito tiene asociado un término con más de un significado como el patrón anterior entonces el requisito está en presencia de ambigüedad léxica.

Detección de ambigüedad Sintáctica

Para comprobar la presencia de ambigüedad sintáctica en un requisito, se defino las siguientes reglas.

R1: (Token)∗ (CC) (Token.kind != "punct")∗ (CC) (Token)∗

R2: (Token)∗ (CD) (Token.kind != "punct")∗ (CD) (Token)∗

R3: (Token)∗ (P) (Token.kind != "punct")∗ (P) (Token)∗

R4: (JJ) (NN | NNS) (y | o) (NN | NNS)

La regla 1 busca al menos dos apariciones de conjunciones copulativas (y, e, ni, que) en el requisito. Estas conjunciones no deben estar separadas por signos de puntuación (por ejemplo, comas, punto y coma, guiones, etc.). Las comas y otros tipos de puntuación pueden aclarar la estructura sintáctica. La regla 2 por su parte busca la

presencia de al menos dos conjunciones disyuntivas (o, u, sea, bien), aunque el funcionamiento es similar a la regla 1. Si un requisito contiene al menos una preposición separable (a, con, de, en) entonces estamos en presencia de la regla 3. Si un adjetivo (JJ) precede a un par de sustantivos singulares (NN) o plurales (NNS), unidos por "y" o "o" estamos en presencia de la regla 4.

Resultados y discusión

Para probar el rendimiento de la herramienta, reunimos un dataset de compilación de requisitos de los centros de la Universidad de Ciencias Informáticas (UCI), además de algunos escenarios de usuario obtenidos internamente. De este modo la herramienta tiene una línea de base para el análisis de los resultados. En primer lugar, procesamos el dataset en la herramienta y luego comprobamos manualmente. A continuación, se muestran los resultados y se concluye en una tabla 3 utilizando tres métricas principales: precisión, exhaustividad y F1.

- Exhaustividad: para informar la **cantidad** que los algoritmos son capaces de detectar ambigüedades.
- Precisión: para medir la **calidad** de los algoritmos de detección de ambigüedades.
- F1: se utiliza para combinar las medidas de precisión y exhaustividad en un sólo valor. Esto es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.

En la tabla 3, se evalúan cada una de las métricas definidas. La evaluación se realiza con un conjunto de 100 requisitos pertenecientes al dataset de requisitos desarrollado por los autores de la investigación. De los 100 requisitos, 94 se detectó correctamente que presentaban ambigüedad y 6 se detectó correctamente que no tenían ambigüedad. La tabla 4 muestra la distribución de los falsos negativos y positivos, además de los verdaderos positivos y negativos.

Tabla 3: Evaluación de SIDARES

Criterios de Medida	**SIDARES**
Precisión	98 %
Exhaustividad	91 %

F1	94 %

Tabla 4: Matriz de confusión

		Precisión	
		0	**1**
Respuesta	**0**	VN	VP
	1	FN	FP

		Precisión	
		0	**1**
Respuesta	**0**	40	54
	1	5	1

0: detecto correctamente que no hay ambigüedad

1: detecto correctamente que hay ambigüedad

VN: verdaderos negativos

VP: verdaderos positivos

FN: falsos negativos

FP: falsos positivos

Fórmulas de las métricas

Precisión = VP **/** VP **+**FP **=** 54 **/** 54 **+** 1 **=** 0.98 **=** 98%

Exhaustividad = VP/ VP **+** FN = 54 / 54 **+** 5 **=** 0.91 **=** 91%

F1=2 **(**Precisión*****Exhaustividad / Precisión+ Exhaustividad**) = 2 (**0.98 ***** 0.91 / 0.98 **+** 0.91**) =** 0.94 **=** 94 %

El **98%** de precisión significa que el algoritmo se equivocara un **2%** de las veces cuando detecte las ambigüedades. El **91%** de exhaustividad nos informa del porcentaje de clases positivas que ha sido capaz de identificar correctamente(verdaderos positivos, verdaderos negativos),es decir la capacidad de detectar ambigüedades o no en los requisitos correctamente .Por último la métrica F1 combina la precisión y la exhaustividad en una sola medida .Estas métricas analizadas en la tabla 3 demuestran que la herramienta obtenida permite detectar ambigüedades léxicas y sintácticas presentes en los requisitos de software.

Conclusiones

El lenguaje natural es el que se suele utilizar para documentar requisitos, por ser un lenguaje intuitivo, expresivo y universal. Dado que el lenguaje natural es generalmente ambiguo, las especificaciones de requisitos deben ser revisadas cuidadosamente para comprobar su calidad lingüística. Para apoyar el trabajo de los ingenieros de requisitos y mejorar la calidad de la etapa de documentación tan importante en la IR, este artículo presenta SIDARES, una herramienta de PLN para detectar ambigüedad léxica y sintáctica en requisitos de software. En el futuro, tenemos previsto mejorar la capacidad de detección de ambigüedades léxicas y sintácticas para reducir los falsos positivos, ya que se sabe que los sistemas basados en reglas. También está previsto incorporar a la herramienta otros algoritmos que permitan detectar otros tipos de ambigüedades como son la semántica y pragmática; para mejorar aún más la herramienta siendo esta más potente al detectar más niveles lingüísticos.

Referencias

1. Ambriola Vincenzo, Gervasi Vincenzo.2018. Processing Natural Language Requirements.
2. Arendse Brian,Lucassen Gram.2016.Toward tool Mashups:Comparing and Combinig NLPRE tools.
3. Casamayor Agustin, Godoy Daniela, Campo Marcelo.2018. Functional grouping of natural language requirements for assistance in architectural software design. 78-86.VOL 30.DOI 10.1016/j.knosys.2011.12.009.
4. Falessi Davide, Cantone Giovanni, Canfora Gerardo.2018. Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. IEEE Transactions on Software Engineering. 18-44.VOL 39.DOI 10.1109/TSE.2011.122.
5. Ferrari, Alessio, Giorgio O. Spagnolo, Antonella Fiscella, y Guido Parente. 2019. «QuOD: An NLP Tool to Improve the Quality of Business Process Descriptions». En Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11865 LNCS:267-81. Springer Verlag. https://doi.org/10.1007/978-3-030-30985-5_17.
6. Gnesi, Stefania, y Gianluca Trentanni. 2019. «QuARS a NLP Tool for Requirements Analysis». http://quars.isti.cnr.it/.

7. Guzmán-Luna, J.A, Gómez Arias, y & Vélez-Carvajal. 2018. «Un modelo de procesamiento de lenguaje natural para la detección de errores en requisitos de software».
8. Hayman Oo, Khin, Azlin Nordin, Amelia Ritahani Ismail, y Suriani Sulaiman. 2018. «An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS)». International Journal of Engineering & Technology. Vol. 7. www.sciencepubco.com/index.php/IJET.
9. Kumar Gupta, Ashok, Aziz Deraman, y Shams Tabrez Siddiqui. 2019. «A Survey Of Software Requirements Specification Ambiguity» 14 (17). www.arpnjournals.com.
10. López Natalia.2019. La interpretación subjetiva de la ambigüedad léxica: una aplicación lexicográfica.
11. Morales-Ramirez, Itzel, Fitsum Meshesha Kifetew, y Anna Perini. 2019. «Speech-Acts Based Analysis for Requirements Discovery from Online Discussions». Information Systems 86 (diciembre): 94-112. https://doi.org/10.1016/j.is.2018.08.003.
12. Naeem, Afrah, Zeeshan Aslam, y Ali Shah. 2019. «Analyzing Quality of Software Requirements; A Comparison Study on NLP Tools».
13. Ortuño Sánchez, Yamila, Yordanis, García Leiva, Yarina, Amoroso Fernández, Reinier, y Silverio Figueroa. 2016. «Herramienta informática para la identificación de la ambigüedad en textos de la legislación cubana.
14. Osama, Mohamed, Aya Zaki-Ismail, Mohamed Abdelrazek, John Grundy, y Amani Ibrahim. 2020. «Score-Based Automatic Detection and Resolution of Syntactic Ambiguity in Natural Language Requirements». En Proceedings - 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME 2020, 651-61. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICSME46990.2020.00067.
15. Ramírez, Denniye Hinestroza, y Juan Manuel Cárdenas. 2018. «El Machine Learning A Través De Los Tiempos, Y Los Aportes A La Humanidad», 17.
16. Ricardo Javier Celi Parraga, Eleanor Alexandra Varela-Tapia, Iván Leonel Acosta Guzmán, Nestor Rafael Montaño Pulzara. 2021. « Técnicas de procesamiento de lenguaje natural en la inteligencia artificial conversacional textual».
17. Soren Lauesen. 2020. Requirements Engineering: Foundation For Software Quality: 18th International Working Conference.
18. Tjong Fatimah, Berry Daniel M.2018. LNCS 7830 - The Design of SREE — A Prototype Potential Ambiguity Finder for Requirements Specifications and Lessons Learned.

19. Zhao, Liping, Waad Alhoshan, Alessio Ferrari, y Keletso J. Letsholo. 2022. «Classification of Natural Language Processing Techniques for Requirements Engineering», abril. http://arxiv.org/abs/2204.04282.
20. Zhao, Liping, Waad Alhoshan, Alessio Ferrari, Keletso J. Letsholo, Muideen A. Ajagbe, Erol Valeriu Chioasca, y Riza T. Batista-Navarro. 2021. «Natural Language Processing for Requirements Engineering». ACM Computing Surveys 54 (3). https://doi.org/10.1145/3444689.