UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Generación del lenguaje del dominio de aplicación a partir de conversaciones informales

AUTORES: Juan Altamirano

DIRECTOR: Leandro Antonelli

CODIRECTOR: Gustavo Rossi

CARRERA: Licenciatura en Sistemas

Resumen

Comprender el contexto de un sistema de software durante la especificación de requerimientos es una tarea difícil y crítica. Definir el lenguaje del dominio antes de especificar los requerimientos es una manera de hacer frente a este problema. El LEL es un documento que permite definir el lenguaje de un dominio particular. Como se ha investigado, el uso de este documento es de gran apoyo para todas las etapas de la construcción de software, sin embargo, el esfuerzo que se debe hacer para construirlo impide que sea una técnica de uso habitual por los analistas. Esta es la razón por la que se propone una herramienta para automatizar su construcción a partir del procesamiento de las conversaciones entre los analistas y los stakeholders.

Palabras Clave

Léxico Extendido del Lenguaje, Procesamiento de Lenguaje Natural, Chats

Conclusiones

A partir de la investigación y el desarrollo de la herramienta, se concluyó que las capacidades de modelado de reglas junto al uso de técnicas de procesamiento de lenguaje natural permiten a los usuarios extraer el documento LEL con poco esfuerzo. La herramienta desarrollada presenta una estrategia para capturar el LEL inédita por la libertad de modelado de reglas que provee a los usuarios.

Trabajos Realizados

Se implementó una herramienta que permite a los usuarios modelar reglas que definen la forma en que serán procesados los textos, para extraer símbolos, nociones e impactos. Esto es posible mediante el uso de técnicas de procesamiento de lenguaje natural. Se desarrollo una sala de chat de ejemplo, para conectar la herramienta, y generar los datos de entrada. Además, se utilizó un framework web para proveer una interfaz intuitiva al usuario, para administrar tanto su documento LEL, como las reglas que haya modelado.

Trabajos Futuros

Se proponen mejoras tales como: Construir un adaptador para conectar la herramienta a otras fuentes de datos de uso masivo (Chats de whatsapp, facebook, etc), agregar el uso de dependencias gramaticales a las condiciones de las reglas y generar una UI para testear las reglas durante su modelado.

Fecha de la presentación: Febrero 2023

Generación del lenguaje del dominio de aplicación a partir de conversaciones informales

Juan Altamirano

10 de febrero de 2023

Resumen

Es importante que al iniciar el desarrollo de software los requerimientos necesarios sean lo más completos y correctos posibles. Comprender el contexto de un sistema de software durante la especificación de requerimientos es una tarea difícil y crítica. Todo aquello que no sea detectado durante esta etapa, o resulte mal entendido, provocará un impacto negativo en los requerimientos, propagando estos errores a las etapas sucesivas del desarrollo de software.

Definir el lenguaje del dominio antes de especificar los requerimientos es una manera de hacer frente a este problema. El LEL es un documento que permite definir el lenguaje de un dominio particular. Como se ha investigado, el uso de este documento es de gran apoyo para todas las etapas de la construcción de software, sin embargo, el esfuerzo que se debe hacer para construirlo impide que sea una técnica de uso habitual por los analistas.

Esta es la razón por la que se propone una herramienta para automatizar su construcción a partir del procesamiento de las conversaciones entre los analistas y los stakeholders.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este primer capítulo se plantean la motivación y los objetivos de esta tesis, para terminar con una explicación general de la distribución de los temas tratados en cada capítulo.

1.1. Motivación

El desarrollo de software es una actividad compleja que requiere la participación de distintos actores con diferentes backgrounds. Durante el desarrollo ellos realizan varias tareas en diferentes momentos, con el objetivo de construir diversos productos de software. Esto significa que es difícil definir una planificación precisa para organizar el desarrollo al comienzo del proyecto, cuando la información sobre este no es muy detallada. El resultado es una planificación de grano grueso cuyas fechas límite se convierten en objetivos a cumplir en vez de estimaciones razonables de entrega del producto. Como consecuencia, resolver un delay se hace prácticamente imposible. En la ingeniería de software, en general, las tareas son muy específicas, y si añadimos más personas para sobrellevar el delay, terminamos perdiendo más tiempo (por lo menos en el inicio, cuando las personas nuevas son incorporadas) [7].

Los requerimientos en la Ingeniería de Software son las descripciones de qué es lo que el sistema debe hacer (los servicios que provee y las restricciones en su operación). Estos requerimientos reflejan las necesidades de los clientes por un sistema que sirve a un cierto propósito, como controlar un dispositivo, buscar información, etc. El proceso de encontrar, analizar, documentar y validar estos servicios y limitaciones se llama ingeniería de requerimientos. Davis[17] sugiere que los errores introducidos durante las actividades de la ingeniería de requerimientos son los culpables del 40 % al 50 % de todos los

defectos encontrados en un producto de software.

El proceso de desarrollo de software se puede representar como una sucesión de descripciones en diferentes lenguajes, donde la descripción anterior es necesaria para producir la siguiente [37]. Entonces, si se introducen cambios en una descripción, las anteriores y las siguientes descripciones deben ser modificadas para mantener la conformidad. Boehm [4] dice que si se produce un error en la descripción de requerimientos y esto es corregido en la etapa de codificación, el costo de la corrección podría ser multiplicado por hasta 200. Más aún, Mizuno desarrolló la "catarata de errores" [31], en donde él asegura que en cada etapa del desarrollo de software la posibilidad de ocurrir errores es mayor que en la etapa anterior, por que cada etapa depende de productos definidos en etapas anteriores. Por lo tanto, es importante comenzar el desarrollo de software con requerimientos que sean tan correctos y completos como sea posible. Aunque estos dos atributos son necesarios en la definición de los requerimientos, tenemos que encontrar maneras de disminuir los conflictos que puedan suceder en el contexto de los requerimientos. Definiendo el lenguaje del dominio antes de especificar los requerimientos puede ser una forma de tratar este problema. Los stakeholders manejan un lenguaje distinto del lenguaje del equipo de desarrollo. Los stakeholders manejan el lenguaje natural, mientras que los miembros del equipo de desarrollo necesitan descripciones más precisas y formales. Es difícil lograr especificaciones de requerimientos que sean adecuadas y útiles para las dos partes, por que el lenguaje natural carece de la precisión y formalidad necesaria, mientras que los usuarios no están familiarizados con los lenguajes formales.

Que el usuario y el desarrollador compartan el mismo lenguaje asegura la comunicación entre ambos. En particular, el uso del lenguaje propio del usuario mejora considerablemente esta comunicación. En el proceso de Ingeniería de Requerimientos, la validación de los diferentes productos requiere una fuerte interacción con el usuario, la que se ve facilitada por el vocabulario común usuario-desarrollador. Las diferentes representaciones que se construyen en el proceso de desarrollo de software, encuentran en el vocabulario del usuario, un marco referencial que permite mantener la continuidad del vocabulario. Lo que a su vez facilita el acceso a la documentación por parte de todos los participantes en el desarrollo.

El vocabulario del cliente puede ser definido mediante un glosario, esto no es más que un diccionario especifico para ese dominio. Cox [14] muestra la importancia de contar con glosarios, en su estudio de la guía de buenas practicas de la ingeniería de requerimientos.

El Léxico Extendido del Lenguaje (Language Extended Lexicon, LEL)

es un glosario que permite describir el lenguaje del dominio de la aplicación utilizando el lenguaje natural. Su objetivo es describir ciertas palabras o frases (símbolos) peculiares a la aplicación y que su comprensión son vitales para poder comprender el contexto de la misma. El LEL no brinda una descripción de requerimientos, sino que sólo permite describir el lenguaje del dominio, a pesar de que hay trabajos que muestran cómo identificar requerimientos a partir del LEL [25].

El LEL es una herramienta muy conveniente para expertos sin habilidades técnicas, sin embargo, las personas con tales habilidades pueden obtener un mayor beneficio con su uso. La conveniencia de la utilización de LEL proviene de 3 características significativas: es fácil de aprender, es fácil de usar y posee buena expresividad, como lo validan experiencias en dominios complejos. Gil [19] indica que: "la experiencia de construir un LEL de una aplicación completamente desconocida para los ingenieros de requerimientos y con un lenguaje altamente complejo, puede ser considerada exitosa, desde el momento en que los usuarios fueron los que notaron que los ingenieros de requerimientos habían desarrollado un gran conocimiento sobre la aplicación". Por su parte, Cysneiros [15] indica que: "el uso del LEL fue muy bien aceptado y comprendido por los stakeholders. Dado que los stakeholders no eran expertos en los dominios tan complejos y específicos en los que trabajaron, los autores creen que el LEL puede ser adecuado para utilizarse en muchos dominios".

Otro objetivo de la utilización del LEL, es la posibilidad de utilizarlo como herramienta para la capacitación de los miembros del equipo de desarrollo que se incorporan en etapas avanzadas del desarrollo, basado esto en una documentación consistente de los símbolos que representan el lenguaje de la aplicación.

Leite et al. [Leite89] proponen construir el LEL como primer paso en la fase de elicitación de los requerimientos: "despreocuparse durante este procedimiento de entender el problema y abocarse sólo a conocer el vocabulario que utiliza el usuario en su mundo real". Una vez familiarizados con ese léxico, el camino para comunicarse con el usuario y comprender el problema tendrá un obstáculo menos: el vocabulario.

La construcción del LEL se realiza mediante un proceso iterativo que involucra a los analistas y a los principales actores del dominio del lenguaje a definir. Primero, los analistas recopilan información sobre el dominio, ya sea documentos o entrevistando a los stakeholders. De la lectura de esa información se obtiene un listado preliminar de símbolos. Luego, dichos símbolos deben ser clasificados y definidos. Los analistas deben evaluar las descripciones de los símbolos, y en caso de haber dudas, recurrir a los stakeholders

nuevamente. Esto implica que el listado de símbolos y sus definiciones puede cambiar, requiriendo hacer dicha evaluación más de una vez. Por ultimo, los analistas deben controlar la estructura del LEL, buscando símbolos sin definir, mal clasificados o con definiciones no acordes a su clasificación. El proceso de construcción del LEL no es complejo, pero si requiere de esfuerzo, ya sea de un único analista consolidando todo el conocimiento, o de un equipo que colabore para generarlo.

El esfuerzo de construcción del LEL se ve agravado por la falta de herramientas que simplifiquen su construcción, requiriendo a los analistas el uso de herramientas no especificas. Existen limitaciones en el uso de enfoques basados en documentos manuales que se corrigen con herramientas específicas, como indica Wiegers [23]: "Herramientas de ingeniería de requerimientos son usadas por analistas de negocio para trabajar con stakeholders para elicitar y documentar requerimientos más efectivamente y eficientemente que con métodos manuales". Herramientas de carácter general, como procesadores de textos, ayudan en la construcción del LEL, sin embargo carecen de soporte de los principios básicos de la definición del LEL, y de las buenas prácticas [24] necesarias para su correcta construcción e interpretación.

De esta manera, queda al descubierto la necesidad de definir un método y una herramienta para simplificar su construcción.

Actualmente, existen herramientas que facilitan la construcción del LEL, como en [34] donde se construyó una aplicación web colaborativa. Sin embargo, disponer de una herramienta específica para la construcción del LEL no resultaría práctica de incorporar al conjunto de herramientas que ya se utilizan durante el análisis y desarrollo del software.

Se han propuesto diferentes métodos para construir el LEL, como en [41] donde se propone derivarlo de las historias de usuario. Sin embargo, este enfoque precisa que la información este estructurada para ser procesada, lo que implica un esfuerzo adicional.

Como el LEL intenta describir el lenguaje del dominio de la aplicación, una forma de hacerlo podría ser analizando automáticamente el lenguaje que se utiliza mientras los stakeholders se comunican. Actualmente existen muchas aplicaciones que permiten comunicarse, estas aplicaciones también son utilizadas por los equipos de desarrollo y los diversos stakeholders. Toda esa información, generada a partir de conversaciones, supondría una fuente para construir el LEL.

1.2. Objetivos

Esta tesis describe una herramienta que procesa las conversaciones informales de los stakeholders y del equipo de desarrollo, para construir el LEL sin necesidad de intervención humana. Para probarlo, se desarrolló un chat básico, que se integra con la herramienta, y es el punto de partida para la generación del LEL.

Se realizó la construcción de una herramienta, la cual es una aplicación Web. Ésta permitió generar el LEL a partir de la ejecución de reglas que extraen los símbolos de los textos procesados, aplicando técnicas de procesamiento de lenguaje natural. Además, tiene una GUI para administrar el LEL generado y las reglas. Esta herramienta dispone de una interfaz REST que se utiliza para conectarla a un pequeño chat, donde un chatbot asiste a los participantes del chat y los abstrae del proceso de generación del LEL.

1.3. Estructura de la tesina

Explicada la motivación y los objetivos abordados por esta tesina, en el siguiente capitulo se describen los temas necesarios para comprender el dominio de la tesina, esto es el Léxico Extendido del Lenguaje y técnicas de Procesamiento de Lenguaje Natural (PLN). En el capítulo 3 se describe la investigación realizada de herramientas similares a la desarrollada en esta tesina. En el capítulo 4 se describe la funcionalidad y la arquitectura de la herramienta construida. Luego, en el capítulo 5 se describen los resultados obtenidos de analizar un caso de estudio. En el capítulo 6 se realiza la evaluación de la usabilidad de la herramienta. En el capítulo 7 se presenta el manual de uso. Finalmente, en el capítulo 8 se presentan las conclusiones y se dejan proyectadas algunas posibles líneas de investigación futuras.

Capítulo 2

Background

Para comprender como se resolvieron los requerimientos de esta tesina es necesario que el lector se familiarice con diversos temas. El objetivo de este capítulo es describir estos temas.

Este capítulo está compuesto por las siguientes secciones: la sección 2.1 describe que es el Léxico Extendido del Lenguaje y cuales son los pasos para construirlo. En la sección 2.2 se describen las técnicas de Procesamiento de Lenguaje Natural utilizadas para implementar la herramienta LELGenerator.

2.1. Léxico Extendido del Lenguaje

El Léxico Extendido del Lenguaje (LEL) [28] es un glosario que tiene por finalidad describir el lenguaje del contexto de la aplicación. Su objetivo es el describir ciertas palabras o frases peculiares a la aplicación y que su comprensión son vitales para poder comprender el contexto de la misma. LEL está anclado en una idea simple "entender el lenguaje del problema sin preocuparse por entender el problema". De esta forma, luego de comprender el lenguaje, el analista podrá escribir requerimientos utilizando como base el conocimiento adquirido a través del lenguaje capturado.

El LEL es un glosario en el cual se definen símbolos (términos o frases). A diferencia del diccionario tradicional que posee sólo un tipo de definición por término (puede haber muchas acepciones, sin embargo, todas son significados del término que se está describiendo). En el LEL cada símbolo se

define a través de dos atributos: la noción (notion) y los impactos (behavioural responses). La noción describe la denotación, es decir, describe las características intrínsecas y sustanciales del símbolo. Por su parte, los impactos describen la connotación, es decir, un valor secundario que adopta por asociación con un significado estricto.

La Figura 2.1 muestra el Modelo del Léxico Extendido del Lenguaje.

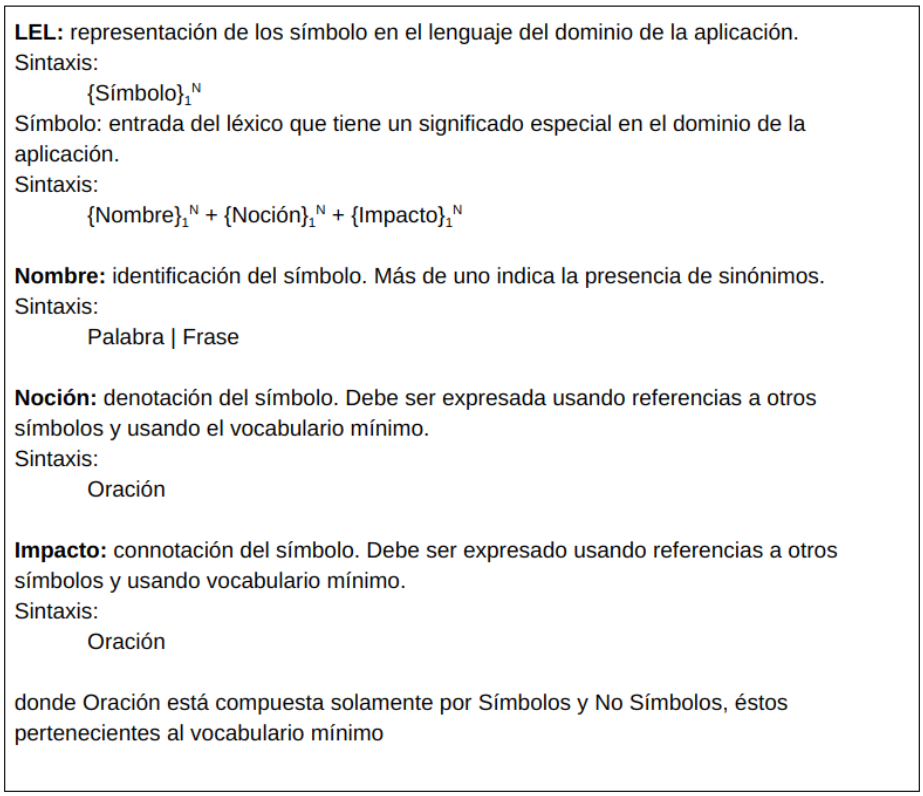

Figura 2.1: Modelo del Léxico Extendido del Lenguaje.

Los símbolos se deben categorizar en una de cuatro categorías básicas con el fin de especializar la descripción de los atributos. En principio hay 3 ontologías básicas que permiten modelar el mundo [5]. Ellas son: entidades, actividades y afirmaciones. Con estos 3 elementos se puede describir al mundo, puesto que permiten modelar las cosas, las actividades con las

que interactúan las cosas y finalmente las afirmaciones o verdades que las cosas o actividades poseen o persiguen. Para el LEL se determinan cuatro categorías básicas que son una extensión de las tres ontologías. Las cuatro categorías básicas son: sujeto, objeto, verbo y estado. Tanto los sujetos como los objetos son una especialización de las entidades. Luego, las actividades se corresponden con los verbos. Y finalmente las afirmaciones se corresponden con estados. Los sujetos se corresponden con elementos activos dentro del contexto de la aplicación, mientras que los objetos se corresponden con elementos pasivos. Por su parte, los verbos son las acciones que realizan los sujetos utilizando los objetos. Finalmente, los estados representan las situaciones en las que se pueden encontrar los sujetos o los objetos previo y luego de realizar las acciones. En la figura 2.2 se presentan las cuatro categorías de símbolos y cómo se debe realizar la descripción de los mismos según sus atributos.

Categoría	Características	Noción	Comportamiento
Sujeto	Elementos activos que realizan acciones	Características o condiciones que los sujetos satisfacen	Acciones que el sujeto realiza
Objeto	Elemento pasivo con los cuales se realizan acciones que puede pasar por distintos estados	Características o atributos que los objetos poseen	Acciones que son realizadas con los objetos
Verbo	Acción que realiza un sujeto, servicio que brinda un objeto desencadenante para pasar de un estado a otro	Objetivo que el verbo persigue.	Pasos necesarios para completar la acción
Estado	Situación en la que se encuentra un sujeto o estado	Situación que representa el estado	Acciones necesarias para pasar a un estado previo o subsecuente.

Figura 2.2: Descripción de los símbolos y sus atributos de acuerdo a su categoría

Existen dos principios que se deben seguir al describir símbolos: el principio de circularidad (también llamado principio de cierre o clausura) y el principio de vocabulario mínimo. El principio de circularidad establece que durante la descripción de los símbolos se debe maximizar el uso de otros símbolos descritos en el LEL. Por su parte, el principio de vocabulario mínimo complementa el principio de circularidad y establece que en las descripciones se debe minimizar el uso de símbolos externos al LEL y los que se

utilicen deben tener una definición clara, unívoca y no ambigua. Ambos principios son vitalmente importantes para obtener un LEL autocontenido y altamente conectado. Estas características redundan en beneficios para comprender el lenguaje de la aplicación, y también hacen que el LEL pueda ser visto como un grafo, el cual, al contener nodos con información textual determina que el mismo sea un hipertexto.

2.1.1. Proceso de construcción del LEL

La construcción de un LEL comienza por obtener información del dominio. A partir de esta información se elabora una lista de símbolos que se deben conocer para entender el lenguaje del dominio. Estos símbolos se deben clasificar para poder definirlos en forma consistente. Luego de clasificarlos, se los define y como producto de la definición se pueden descubrir sinónimos, por lo cual se deben reorganizar los símbolos. La información debe ser validada por los expertos del dominio y controlada por el ingeniero en requerimientos. Si algún nuevo símbolo debe ser definido, se repite el proceso [27][3].

El proceso de construcción es descrito en la figura 2.3.

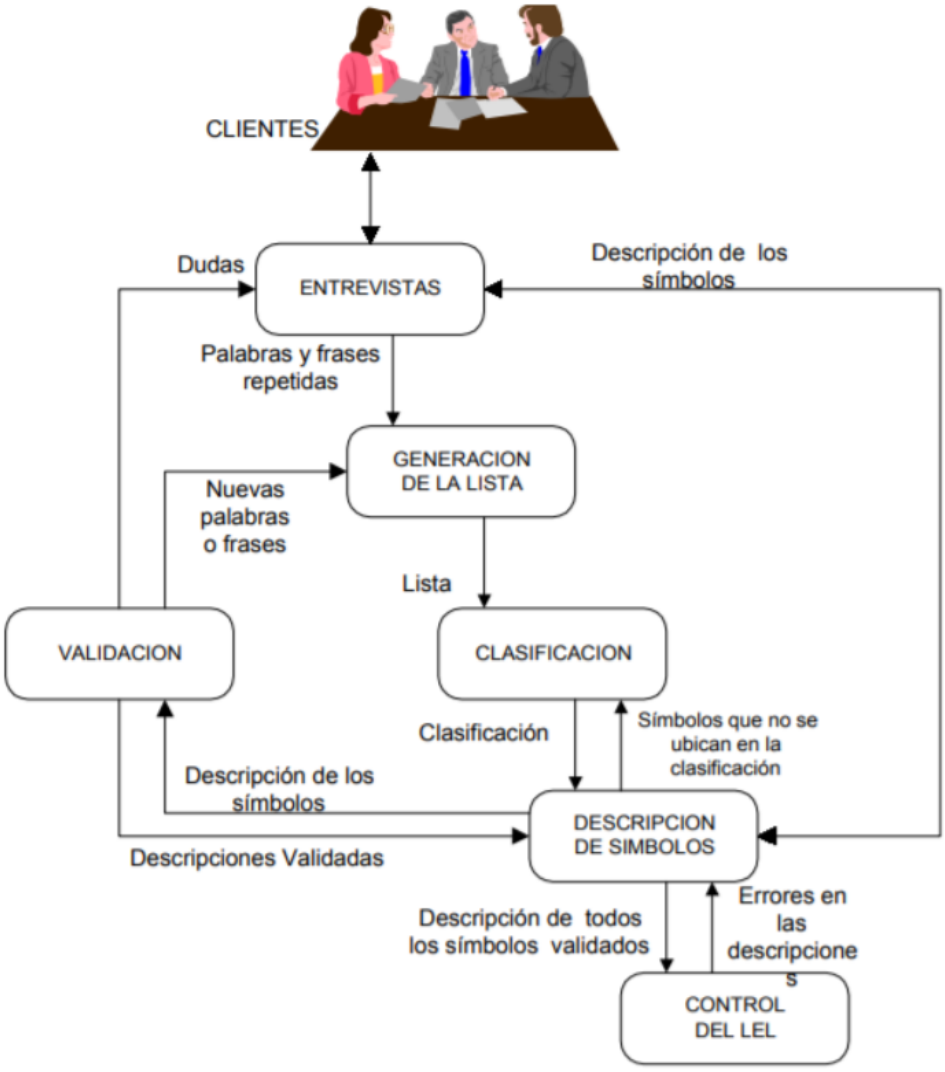

Figura 2.3: Proceso de construcción del LEL

Identificación de las fuentes de información

Las fuentes de información para la construcción del LEL son dos: las personas y los documentos. Cada uno tiene ventajas sobre el otro. Los documentos ofrecen un medio concreto, auto contenido e invariable.

El ingeniero de requerimientos puede leer y volver a leer los documentos para una selección minuciosa de los términos. Los textos se pueden analizar en detalle. Por otra parte, las personas pueden aportar a través de entrevistas mucha información. Una característica propia de la expresión oral de los seres humanos es que en forma inconsciente utilizan palabras propias del dominio, es decir aplican el principio de circularidad. En cambio, en la descripción escrita para una narrativa más entretenida se introduce sinónimos, lo que puede dificultar el proceso de identificación de símbolos. Es aconsejable utilizar ambas fuentes de información.

Las personas que son aconsejables entrevistar son:

- Las más mencionadas dentro del entorno de sistema.
- Las que toman decisiones.
- Los responsables del proyecto.
- Los supervisores del proceso.

Por su parte, los documentos que más información brindan son:

- Descripciones textuales del sistema
- Manuales que definen los procedimientos operacionales.

Generación de la lista de símbolos

Una estrategia aconsejable para generar la lista de símbolos consiste en comenzar con la lectura de la documentación para hacer un análisis preliminar del lenguaje del dominio. Con este análisis se obtiene una lista inicial y se adquiere información para poder llevar a cabo entrevistas. Es aconsejable realizar entrevistas semi estructuradas y estructuradas para acotar el lenguaje, debido a que las entrevistas libres conducen a un gran volumen de información lo que se hace muy difícil de manejar.

En esta lista inicial se deben tomar aquellos símbolos o frases que se utilizan con mucha frecuencia, aquellas a las que se les hace énfasis (aparecen en negrita, cursiva o subrayado) o aquellas que parecen estar fuera de contexto. El motivo de esta elección de símbolos o frases radica en el objetivo del LEL,

el cual consiste en capturar el lenguaje de un dominio. Los términos más utilizados son significativos del dominio, por lo que deben estar definidos. En cambio los términos que parezcan estar fuera de contexto, es debido a que tienen un significado propio en el dominio, distinto del tradicional. Estos términos, con más razón, deben estar definidos.

Cabe destacar que los símbolos no tienen porqué ser palabras individuales. Pueden ser palabras o frases. La razón es que en un lenguaje se puede encontrar un grupo de palabras, en donde cada una de ellas puede tener cierto significado, pero si se las combina en una frase puede tener otro.

Clasificación de la lista de símbolos

Antes de realizar una descripción de los símbolos, se debe llevar a cabo una clasificación. Esta clasificación consiste simplemente en determinar para cada símbolo si es sujeto, objeto, verbo o estado. La misma tiene como fin el de poder realizar una administración del conjunto de símbolos. Además, dependiendo la categoría a la que pertenezca cada símbolo es la forma en la que debe ser definido. Vale aclarar que esta categorización no es definitiva, sino que es una categorización preliminar y luego en la descripción puede surgir que la categoría determinada no era la correcta.

Descripción de los símbolos

Los símbolos se describen a partir del conocimiento obtenido de la lectura de la documentación y de las entrevistas. Para cada símbolo se deben escribir la noción y los impactos teniendo en cuenta la clasificación establecida en la etapa anterior. Por otra parte, suele ocurrir que la descripción de los símbolos no surja del conocimiento que el analista recopiló, sino que necesita recurrir a las fuentes para completar la descripción. Esto podría llevar a que aparezcan nuevos símbolos que necesitan ser definidos.

Algunas pautas generales para la descripción de los símbolos, independientemente de la clasificación inicial que luego será validada y controlada, son:

- Cada noción e impacto debe ser descrito con oraciones breves y simples.
- Las oraciones deben responder a los principios de circularidad y de vocabulario mínimo.
- Las nociones e impactos de un símbolo pueden representar diferentes puntos de vista o pueden ser complementos.

Validación

Consiste en validar la correctitud del LEL contra el usuario. Debido a la gran cantidad de símbolos que pueden estar definidos en el LEL, es impracticable realizar una validación completa y exhaustiva. Sin embargo, es posible mantener sesiones de entrevistas estructuradas para aclarar dudas.

Control

El proceso de control es el que realiza el ingeniero de requerimientos por sus propios medios. El control no tiene que ver con la correctitud de la información del LEL, por el contrario está relacionado con la estructura:

- Todos los símbolos deben estar definidos. Aquellos de la lista inicial y también los que aparecen en la descripción de los signos de la lista inicial.
- Todos los símbolos deben estar dentro de la clasificación correspondiente. Debido a que la clasificación se realiza antes de la descripción, en esta última se puede descubrir que un símbolo se colocó en la categoría incorrecta, por lo cual se lo debe reubicar y verificar su definición.
- La descripción de los símbolos debe corresponderse con la de la categoría a la que pertenecen.
- El punto de vista para la descripción de los símbolos debe ser uniforme.
- No deben dejarse sinónimos definidos como signos independientes.

2.1.2. Conclusión

Como se pudo ver en esta sección, un documento LEL mejora ampliamente el conocimiento acerca del dominio para todos los stakeholders, a su vez esto influye directamente sobre las especificaciones de requerimiento, ya que con un conocimiento adquirido acerca de una aplicación y su dominio, los requerimientos especificados serán más completos y menos ambiguos. Pero un LEL no solo se utiliza para mejorar el conocimiento del dominio, si no que ya existen técnicas estudiadas acerca de cómo utilizar estrategias de derivación para transformar los símbolos y sus descripciones en requerimientos de software directamente, ya sea una historia de usuario o un caso de uso [25]. También, puede ser usado para estimar el tamaño de una aplicación utilizando como referencia un documento LEL [26]. Por tanto, toma mayor relevancia este trabajo final, ya que al conseguir construir el LEL de manera

automática abrimos el camino a automatizar también sus derivaciones, y así, sumar valor a etapas posteriores de la ingeniería de software.

2.2. Procesamiento de Lenguaje Natural

Se determina al lenguaje como un medio por el cual los humanos logran comunicarse y expresar racionamiento, este medio está sustentado por la asociación de signos con ciertos significados. El lenguaje usa herramientas como la escritura, las señales y la voz para establecer comunicación. Aquí es donde se encuentran dos tipos de lenguaje, el lenguaje natural (LN) y los lenguajes formales. En el lenguaje natural encontramos lo que comúnmente llamamos idiomas como inglés, español, alemán entre otros. Estos lenguajes están en constante crecimiento sin tener en cuenta las reglas que los suceden. Por otro lado, los lenguajes formales que se encuentran enmarcados en campos como la matemática, la lógica o la programación, los cuales están ceñidos rigurosamente a reglas establecidas.

El lenguaje natural es enriquecido por su vocabulario y construcciones, también se establecen características como la flexibilidad, ambigüedad e indeterminación permitiendo la variedad en la interpretación dependiendo de la situación. Esto resulta ventajoso en el momento de efectuar la comunicación humana, pero al momento de enfrentarse al procesamiento computacional dichas características se presentan como un problema ya que dificultan la aplicación de procesos de razonamiento, caracterización y formalización.

El lenguaje natural es transversal a todas las áreas de estudio, y desde hace tiempo que se analiza la idea de procesar dicha información de manera automática. Por ejemplo en la ingeniería de software la gran mayoría de pedidos de requerimientos están escritos en LN, y dichos requerimientos están escritos por una persona no experta. En un flujo de trabajo normal, dichos requisitos tienen que ser analizados por un especialista que tiene que ser capaz de comprender y relacionar los conceptos de un dominio específico para generar los documentos formales de las etapas de la ingeniería de software, o cualquier otro documento necesario para un proyecto.

Un documento LEL surge específicamente del LN y sin tener algún documento técnico de respaldo. El analista encargado de la construcción del LEL debe analizar, interpretar y razonar información escrita en LN por parte de un cliente o los stakeholders de un proyecto, y a partir de ese trabajo construye el documento. Entonces, el procesamiento de lenguaje natural surge de la necesidad de procesar documentos y obtener conocimiento de manera automática. Tomando como base la ingeniería de software, el Procesamiento de Lenguaje Natural es una gran herramienta que puede ayudar a construir automáticamente un documento que se necesite a partir de LN.

2.2.1. Definición

Cuando se habla de procesar un lenguaje, se refiere a la traducción de la versión de un texto desde una lengua a otra [20]. Así, para ejecutar Procesamiento de Lenguaje Natural (PLN) se requiere la transformación del texto en una representación semántica apta para razonar, tomar decisiones y ejecutar tareas específicas. Esta representación se logra por medio de procesos de Parsing o construcción de un árbol de análisis a partir de una gramática [29]. Si la gramática es sintáctica, por medio de dicho árbol de análisis se genera información sobre las categorías gramaticales de las palabras y la función sintáctica asociada (por ejemplo: identificación del sujeto, verbo, predicado y complementos). Mientras que, si la gramática es semántica, el árbol de análisis ya es bastante próximo a la representación lógica que permite el razonamiento y la ejecución.

El PLN como disciplina busca desarrollar programas computacionales que sean capaces de ejecutar actividades relacionadas con la comprensión, análisis y producción de textos o discursos escritos en LN, de una manera similar a como lo hace el ser humano [1].

Las aplicaciones y usos de PLN se encuentran en muchas de las cosas que empleamos en nuestra vida diaria. Casos en los cuales se usa PLN son los motores de búsqueda en línea como Google, los traductores y resumidores automáticos, los correctores de estilo y ortografía de los procesadores de texto, los reconocedores de voz, entre otros.

2.2.2. Herramientas lingüísticas

El análisis lingüístico de texto se realiza con procesos en capas. Los documentos son divididos en párrafos, los párrafos en oraciones y las oraciones en palabras individuales. Luego las palabras son etiquetadas según las partes de la oración y otras características, antes que la oración sea analizada (sujeta a análisis gramatical).

Por lo tanto los analizadores se construyen típicamente de delimitadores de oraciones, tokenizers, stemmers y etiquetadores gramaticales (POS Tagging). Pero no todas las aplicaciones requieren una suite completa de estas herramientas. Por ejemplo, todos los motores de búsqueda realizan la etapa de tokenización, pero no todas realizan el etiquetado gramatical.

Para poder analizar oraciones de un documento, primero es necesario determinar el alcance de estas oraciones e identificar cómo está compuesta.

Delimitadores de oraciones

Determinar los límites de una oración no es una tarea sencilla, ya que los signos de puntuación que marcan el fin de una oración suelen ser ambiguos. Por ejemplo, un punto puede significar un punto decimal, una abreviación, el final de una oración o una abreviación al final de una oración. Para desambiguar los signos de puntuación, los delimitadores de oraciones utilizan expresiones regulares o reglas de excepción.

Tokenizadores

Los delimitadores de oraciones suelen necesitar de tokenizadores para desambiguar caracteres de puntuación. Tokenizadores (también conocidos como analizadores léxicos o segmentadores de palabras) segmentan una cadena de caracteres en unidades significativas llamadas tokens. Al principio, la tokenización parece una operación bastante directa: un token podría obtenerse de secuencias de caracteres separados por espacios en blanco. Esta simple aproximación puede parecer apropiada para algunas aplicaciones, pero puede no ser precisa.

Por ejemplo, no tener en cuenta los signos de puntuación, como los puntos, comas y guiones. ¿La palabra "Data-base" está compuesta por uno o dos tokens? claramente el número "1,005.98" debería ser un token. ¿Que hay sobre "$1,005.98"?, ¿debería el signo "$" ser parte del token, o debería ser identificado en un token aparte?

Hasta ahora, hemos confiado en los espacios en blanco para indicar el término de una palabra. Pero esto no es siempre el caso. En algunos lenguajes, de hecho en la mayoría de lenguajes de Asia del este, no se usan espacios en blanco entre palabras. Otros lenguajes, como el Alemán o Koreano, utilizan los espacios en blanco, pero permiten la creación dinámica de palabras compuestas. Estas palabras compuestas pueden ser consideradas como una sola palabra, pero en una tarea de análisis de documentos, nos podríamos beneficiar de identificar las partes que componen esas palabras compuestas. Las herramientas de tokenización suelen depender de reglas, máquinas de estados finitos, modelos estáticos, y léxicos para identificar abreviaciones o palabras multi-token.

El proceso de parseo no puede realizarse sin antes hacer un análisis léxico, entonces es necesario que primero se identifiquen las raíces de las formas de cada palabra y determinar su papel en la oración.

Stemming

El Stemming es una técnica para eliminar afijos de una palabra, teniendo como resultado la raíz ≪stem≫ de la palabra. Por ejemplo, la raíz de las palabras agregar o agregando es ≪agreg≫,un buen algoritmo de stemming sabe que el sufijo ≪ando≫ o ≪ar≫ puede eliminarse. Stemming es más comúnmente utilizado por los motores de búsqueda para indexar palabras. En lugar de almacenar todas las formas de una palabra, un motor de búsqueda puede almacenar sólo las raíces obtenidas de un Stemming, lo que reduce el tamaño del índice mientras que aumenta la precisión de recuperación.

Lemmatization

La lematización es un proceso lingüístico que consiste en, dada una forma flexionada, hallar el lema correspondiente. El lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra. Es decir, el lema de una palabra es la palabra que nos encontraríamos como entrada en un diccionario tradicional: singular para sustantivos, masculino singular para adjetivos, infinitivo para verbos. Entonces, a diferencia del Stemming, siempre queda una palabra válida que significa lo mismo o que es el origen de dicha palabra.

Etiquetado gramatical

El etiquetado gramatical (conocido también por su nombre en inglés, part-of-speech tagging o POS tagging) es el proceso de asignar (o etiquetar) a cada una de las palabras de un texto su categoría gramatical. Este proceso se puede realizar de acuerdo con la definición de la palabra o el contexto en que aparece, por ejemplo su relación con las palabras adyacentes en una frase, oración, o en un párrafo. El etiquetado de POS se puede utilizar en aplicaciones de Texto a voz, recuperación de información, análisis, extracción de información, investigación lingüística para corpus y también se puede utilizar como un paso intermedio para tareas de nivel superior de PLN, como análisis, análisis semántico, traducción y muchos más [22], lo que hace que el etiquetado POS sea una función necesaria para cualquier aplicación basada en PLN.

Como se ve en el ejemplo de la figura 2.4, una herramienta a través de distintas técnicas como pueden ser árboles de decisión o redes neuronales etiquetan de acuerdo al contexto de la oración distintas clases gramaticales a las palabras de una sentencia. Dependiendo de la herramienta a veces la nomenclatura con la que etiquetan puede variar, sin embargo existe una

referencia para seguir, que es el Esquema de Universal Dependencies. Universal Dependencies es un framework para tener una anotación consistente de gramática en diferentes idiomas. En el cual contiene etiquetas para generalizar verbos, adjetivos, sustantivos, adverbios, y demás clases que son transversales en la mayoría de los idiomas [18].

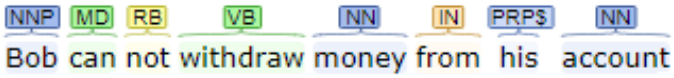

Figura 2.4: Ejemplo de POS tagging

Dependencias gramaticales

A su vez, como hay procesos que realizan una clasificación entre las palabras de una oración, existe una clasificación para las relaciones entre esas palabras. Las cuales son llamadas relaciones gramaticales, las cuales expresan a través de etiquetas, cual es la relación que existe entre dos palabras. Los argumentos de estas relaciones consisten en un elemento de cabecera y un elemento dependiente [22]. Como sucede en el POS Tagging, las relaciones de dependencias también siguen la referencia del Esquema de Universal Dependencies.

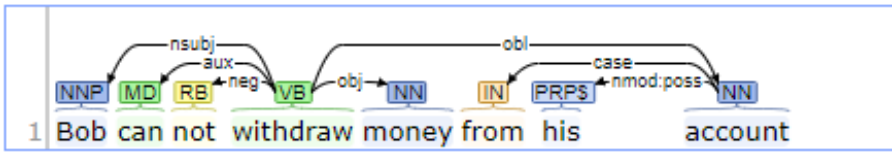

Figura 2.5: Ejemplo de dependencias gramaticales

Como se indica en la figura 2.5, se puede ver como la palabra “Bob”, representada por el tag NNP (nombre propio en singular) es el sujeto nominal (nsubj) del verbo “withdraw”, y la palabra “money” es el objeto principal en la que el verbo interactúa a través de la relación “obj”, que representa el objeto directo.

Coreference resolution

Coreference resolution es la tarea de encontrar todas las expresiones que refieren a la misma entidad en un texto. Es un paso muy importante en muchas tareas de alto nivel de PLN que involucren entender lenguaje natural, como los resumidores de documentos y la extracción de información [10].

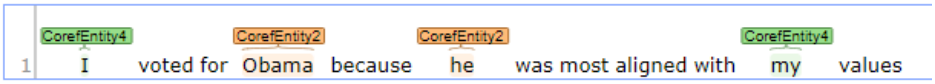

Figura 2.6: Ejemplo de coreference resolution

Como muestra la figura 2.6, se puede ver como se identifican las menciones a las mismas entidades.

Conclusión

La conformación de un documento LEL, que cuenta con sus cuatro respectivos símbolos, sujeto, objeto, verbo y estados, se ve altamente relacionado con las categorías existentes en el lenguaje natural del inglés por lo tanto también se verá reflejado en la manera de clasificar y etiquetar a las palabras por parte del POS Tagging que se puede realizar con las herramientas de PLN. Además, enriqueciendo el análisis del texto utilizando el resto de las técnicas estudiadas, nos da un marco para trabajar y definir las relaciones entre los símbolos dentro de un texto. En base a estas técnicas es que los usuarios deberán definir sus reglas de extracción de símbolos.

Capítulo 3

Estado del arte

Para comprender el marco actual en el que se desarrolla esta tesina y cuales son las contribuciones de esta, primero debemos investigar y analizar que otros trabajos existen que hagan algo similar a lo propuesto. El objetivo de este capitulo es presentar la investigación realizada.

Este capitulo esta compuesto por las siguientes secciones: la sección 3.1 presenta el marco inicial, para luego dar comienzo a la sección 3.2 donde se describen los principales trabajos relacionados a esta tesina. Luego, la sección 3.3 presenta las conclusiones alcanzadas tras realizar las investigaciones y análisis de los trabajos relacionados.

3.1. Introducción

Dentro de los objetivos de la investigación del estado del arte se encontraba buscar trabajos similares al propuesto, y a pesar que el documento LEL es un área de estudio relativamente nueva, hemos encontrado una tesina de grado que lo deriva a partir de lenguaje natural de forma semiautomática [9]. Dicho trabajo se publicó durante la realización de esta tesina de grado, y aunque a priori pareciera ser un trabajo muy similar al propuesto en esta tesina, la estrategia propuesta por su autor difiere en la utilizada en este trabajo. Además, se encontró una herramienta que genera el LEL analizando Historias de Usuario, y aunque se trate de análisis de texto estructurado, describiremos como lograron realizarlo. Por último, se encontró una herramienta web que facilita la construcción del LEL, utilizando características colaborativas. El análisis de estos trabajos encontrados se realizará en la siguiente sección.

3.2. Trabajos relacionados

3.2.1. Construcción del documento LEL a partir de lenguaje natural

En el trabajo "Construcción semiautomática de un documento LEL utilizando técnicas de procesamiento de lenguaje natural" [9] su autor desarrolla una herramienta que efectivamente construye un documento LEL, pero la estrategia aplicada para construirlo difiere a la planteada en este trabajo de grado. Veamos cuales son esas diferencias, y que ventajas y desventajas podemos identificar. Para esto, analicemos los pasos que plantea esta herramienta para resolver el problema:

Datos de entrada

Como primer paso, la herramienta posee una interfaz para que el usuario ingrese todo el texto a analizar. Esto ya representa una gran diferencia respecto a la herramienta que planteamos en este trabajo de grado, ya que nosotros analizamos pequeñas porciones de texto. Luego, se especifica que dicho texto de entrada debe cumplir 2 requerimientos:

- Se debe ingresar un documento que especifique requerimientos funcionales de un sistema.
- El texto debe contar con estructuras sintácticas bien definidas con respecto a sujeto y predicado.

El segundo ítem representa otra diferencia a destacar, ya que en este trabajo no necesitamos que el texto tenga una estructura en particular para ejecutar las reglas modeladas por el usuario sobre éste. Además, el análisis se hace sobre el lenguaje español, mientras que en este trabajo se hace en inglés.

Módulo de reglas heurísticas

Como segundo paso, se ejecutan una serie de reglas que intentan controlar y reestructurar las sentencias que son ingresadas, para generar sentencias de la forma: Sujeto + Verbo + Objeto directo o indirecto. Esto es muy interesante ya que consigue:

- Simplificar el análisis de oraciones en voz pasiva: Es más fácil procesar la oración "Los usuarios compran productos" que "Los productos son

comprados por usuarios", ya que en la segunda oración la herramienta podría confundirse y detectar como sujeto a la palabra "producto".

- Conjunciones de sustantivos y verbos: Pueden ingresarse oraciones como "Los usuarios registrados y no registrados pueden reservar hoteles, autos y pasajes de avión", en este caso es más simple reestructurarlo en las siguientes oraciones: [≪Los usuarios registrados pueden reservar hoteles≫ ≪Los usuarios registrados pueden reservar autos≫ ≪Los usuarios registrados pueden reservar pasajes de avión≫ ≪Los usuarios no registrados pueden reservar hoteles≫ ≪Los usuarios no registrados pueden reservar autos≫ ≪Los usuarios no registrados pueden reservar pasajes de avión≫].

Tiene sentido reestructurar las oraciones si la herramienta esta preparada para procesar textos con formato Sujeto + Verbo + Objeto. Aunque este no sea el caso de la herramienta propuesta en esta tesina de grado, considero que realizar estas mismas transformaciones nos ahorrarían falsos positivos, como veremos en el capítulo 5.

Normalización de la entrada

Luego de la reestructuración de las sentencias a un formato prácticamente único a nivel sintáctico y estructural, el siguiente paso es la normalización convencional que se tiene en la mayoría de los desarrollos de procesamiento de lenguaje natural. El proceso de normalización que se realiza es eliminar las palabras denominadas "stopwords". Estas palabras son aquellas que no agregan una importancia o no cambian la semántica de una sentencia, estos elementos pueden ser pronombre, preposiciones, artículos, etc.

La herramienta propuesta en esta tesina de grado también elimina las "stopwords" del análisis.

Obtener Sujetos y Objetos Principales

Se utiliza la herramienta NLTK y Spacy para aplicar la técnica POS Tagging, la cual permite etiquetar a las palabras con la función sintáctica que cumple en una sentencia. De esta forma se identifican primero los Verbos, para dividir la oración en sujeto y predicado. De los sujetos y predicados se extrae la lista de todos los sustantivos utilizados, para luego aplicar una técnica de análisis estadístico de texto donde se buscan los sustantivos más frecuentes. Esa lista de Sujetos y Objetos es presentada al usuario para que los confirme.

Interacción del usuario con los símbolos

Una vez que el usuario tiene a su disposición la lista de símbolos, el sistema provee una interfaz para que el usuario tenga la posibilidad de poder agregar o eliminar cualquiera de ellos, esto se realiza para tener una retroalimentación del usuario.

Obtener Verbos, Estados y sus nociones

La próxima etapa se centra en hallar los impactos para los sujetos encontrados. Para realizar este proceso se buscan aquellas sentencias donde se encuentre alguno de los sujetos definidos durante la etapa anterior, luego busca los verbos que estén asociados a estos sujetos e identifica sobre que objetos actúan y cuales son los cambios que estas acciones provocan en los objetos también denominados estados. Para que estas sentencias puedan convertirse en un impacto, deberán contar con al menos un sujeto y un objeto del conjunto de símbolos definidos entre el usuario y la herramienta.

Esto implica volver a analizar todo el texto, pero esta vez se utiliza la técnica dependency parsing para encontrar los verbos relacionados a los sujetos y objetos confirmados por el usuario. Antes de presentar al usuario los distintos impactos, se utiliza un modelo de aprendizaje automático para clasificar el texto y determinar si un impacto puede ser válido o no.

Luego, se busca la definición de cada símbolo detectado en la base de datos de Multilengual Central Repository. Por ultimo, el usuario selecciona que información es correcta y cual no lo es, de esta forma se retro-alimenta el clasificador de impactos, para tomar mejores decisiones en la próxima ejecución.

3.2.2. Derivación del documento LEL a partir de Historias de Usuario

Las historias de usuario son descripciones cortas y simples de una característica contada desde la perspectiva de la persona que desea la nueva capacidad, generalmente un usuario o cliente del sistema. Las historias de usuario se ajustan a una plantilla que considera 3 atributos: un rol, un objetivo y un motivo [11]. El rol define el usuario que interactúa con la aplicación para usar la característica descrita por el objetivo. El objetivo representa el requerimiento que la aplicación debe proveer. El motivo explica por que el usuario requiere que la aplicación provea dicha funcionalidad.

Como ***rol*** Quiero ***objetivo*** Para que ***motivo***

En [41] los autores construyeron una herramienta que procesa historias de usuario para obtener el LEL. La estrategia utilizada consiste en procesar un conjunto de reglas que están definidas en dos partes: las condiciones que tienen que cumplirse y el elemento que debe ser incorporado al LEL. De esta manera, por ejemplo, la primera regla determina que todo rol debe ser agregado como un nuevo sujeto. La herramienta utiliza las técnicas de tokenización y etiquetado gramatical para automatizar la ejecución de las reglas. La librería de procesamiento de lenguaje natural utilizada para el desarrollo fue CoreNLP [16] sobre un entorno Java.

3.2.3. Aplicación web para la construcción colaborativa del Léxico Extendido del Lenguaje

En [34] los autores construyen una aplicación web donde los usuarios pueden incorporar los símbolos, para que luego los demás usuarios de la herramienta los valoren de forma positiva o negativa. De esta forma consiguen construir el LEL con la colaboración de los propios usuarios. Por ultimo, los usuarios pueden consultar y administrar los símbolos desde una interfaz desarrollada en Angular. Aunque a priori no se utilizaron técnicas de procesamiento de lenguaje natural, como conclusiones del trabajo, los autores concluyen que se aportarían grandes beneficios en aplicarlas. En particular para detectarlos automáticamente y normalizar la forma en que son almacenados (lemmatization).

3.3. Conclusiones

Es innegable las ventajas que aportan las técnicas de procesamiento de lenguaje natural en la construcción automática de un documento como el LEL. Tras la investigación podemos concluir que no es sencillo procesar texto natural sin estructura formal, y que en estos casos es necesario utilizar técnicas más complejas de PLN. Además, la estrategia a aplicar depende íntimamente del texto de entrada, no solo por su formalidad, sino también en su extensión. En el caso de esta tesina de grado, el trabajo se vuelve único ya que debe procesar los mensajes de un chat a medida que estos son enviados. Además, dada la posibilidad de falsos positivos, es necesario que la herramienta provea la manera en que los usuarios puedan administrar el LEL generado.

Capítulo 4

Herramienta: LELGenerator

La herramienta LELGenerator permite capturar el LEL procesando texto en lenguaje informal. Para esto los usuarios definen un conjunto de reglas a utilizar, esto es un conjunto de condiciones que de evaluar satisfactoriamente sobre el texto, ejecuta un conjunto de acciones sobre el LEL.

Este capitulo esta compuesto por las siguientes secciones: la sección 4.1 describe como esta compuesta la herramienta y que tecnologías se emplearon. En la sección 4.2 se describen las herramientas utilizadas. Luego, en la sección 4.3 se describe el diseño de las reglas que permite modelar y ejecutar la herramienta, y se describe el set inicial de reglas provisto al usuario. Por ultimo, en la sección 4.4 se describe como son ejecutadas las reglas.

4.1. Arquitectura

El desarrollo de la herramienta se basó en el concepto de Arquitectura Orientada a Servicios de Clientes (Service Oriented Architecture, SOA).

SOA es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos de negocios. La utilización de una arquitectura SOA permite que los sistemas sean altamente escalables brindando a su vez una forma bien definida de exposición e invocación de servicios (comúnmente, pero no exclusivamente servicios web).

En particular, para la construcción de la herramienta se utilizó una arquitectura de n-capas, utilizando el lenguaje Java como la tecnología dominante para el soporte de los servicios, siguiendo un patrón de diseño MVC (Model View Controller).

La separación en capas propuesta permite aislar los distintos aspectos del desarrollo de la herramienta, donde las capas inferiores proveen servicios a su capa inmediata superior. Esta forma de separación, además de lo descrito y mediante el uso de interfaces, permite que las capas puedan reemplazarse teniendo un costo e impacto mínimo respecto de las capas con las que se comunica. Un ejemplo de esto puede ser que la capa de servicios de negocios prepara los datos para entregarlos a la capa de presentación. Esta capa se encarga de cómo mostrar los datos que recibe, sin preocuparse de qué datos muestra.

Para describir la arquitectura tomamos el modelo 4C [8] que propone 4 niveles de abstracción, desde una vista general del sistema y su contexto hasta el diagrama de clases de un componente del sistema.

4.1.1. Nivel 1: Contexto

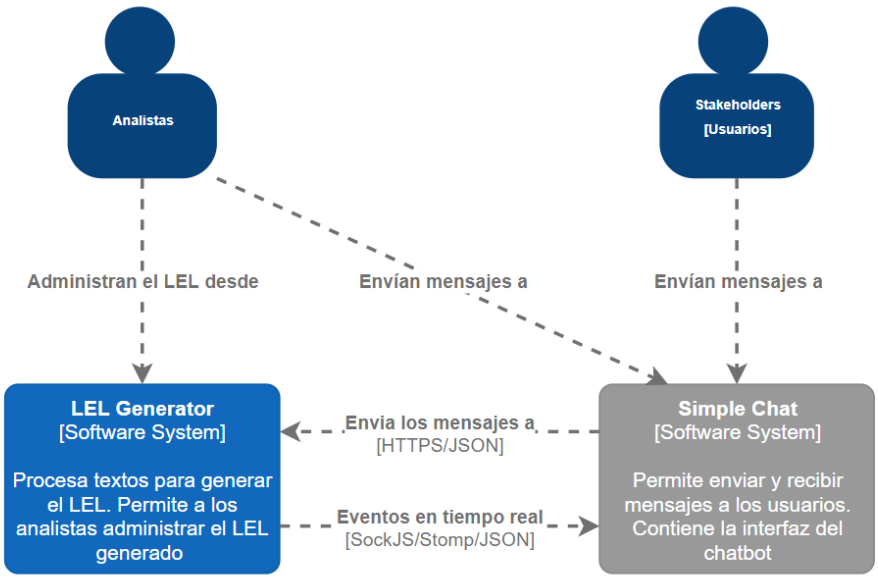

Figura 4.1: Diagrama del contexto de la herramienta LELGenerator

La herramienta LELGenerator es un sistema que provee una interfaz de comunicación vía HTTPS con otros sistemas, para recibir los textos que debe procesar.

Los usuarios analistas se conectan a la herramienta LELGenerator para poder administrar su LEL, esto lo hacen vía web browser.

Además del sistema LELGenerator se describe un segundo sistema llamado "Simple Chat", donde los usuarios entran a chatear. Este será el generador de texto a procesar. Al tratarse de un chat, se requiere que todos los usuarios reciban los mensajes que generan el resto de los usuarios en tiempo real, para esto se utilizó websockets, en particular SockJS [36] y Stomp [38]. Además, la interfaz del chatbot que asiste al usuario está en el chat, y necesita conocer cada vez que la herramienta LELGenerator detecte un nuevo símbolo, noción o impacto.

Lo importante de este diagrama es que el "Simple Chat" puede ser reemplazado por un adaptador para procesar el texto desde otras herramientas. Si se conoce como deben comunicarse, solo debe hacerse esa adaptación, por lo que consideramos que el esquema elegido es lo suficientemente flexible para ser extendido con facilidad.

Lo antes descrito puede verse en la figura 4.1. Ahora que entendemos el contexto en el cual vive nuestro sistema, veamos como esta compuesto, con un grado menor de abstracción.

4.1.2. Nivel 2: Contenedores del sistema LELGenerator

La herramienta LELGenerator está compuesta por 3 grandes contenedores: Una aplicación Single-Page, la lógica de negocio detrás de una api y el esquema en la base de datos.

A continuación se describe cada contenedor especificando la tecnología empleada:

Single-Page App

Una Single-Page App (SPA) es una aplicación web o website que interactúa con el web browser reescribiendo dinámicamente la página web actual con nueva información obtenida del web server, en vez del método tradicional que carga la pantalla completamente [35]. El objetivo es mejorar la velocidad de las transiciones, cada vez que el usuario interactúe con la página. En una SPA, todos los recursos necesarios como HTML, JavaScript y CSS se pueden cargar tanto con una única carga inicial al entrar a la página web, o dinámicamente a medida que los recursos son requeridos. La página

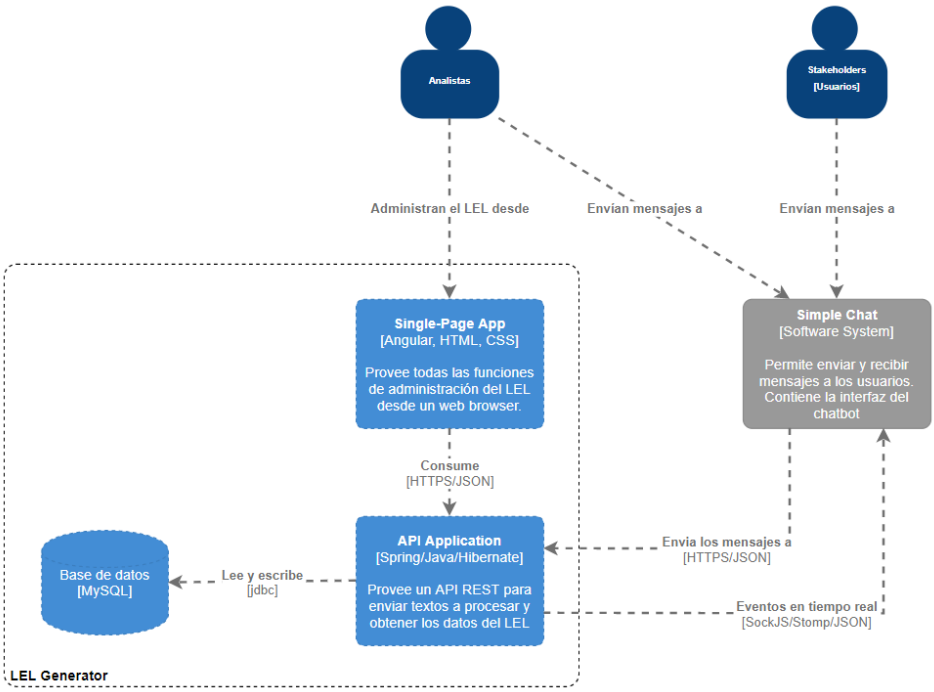

Figura 4.2: Diagrama de contenedores de la herramienta LELGenerator

web no se recargará en ningún otro punto, ni transferirá el control a otra página, aunque la localización o el HTML5 History API pueden ser usados para dar la percepción y navegabilidad de estar trabajando con múltiples páginas diferentes dentro de la misma aplicación.

Existen varios frameworks JavaScript para construir SPA's, para realizar el trabajo se optó por Angular. Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener SPAs [2]. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de MVC, en un esfuerzo por hacer que el desarrollo y las pruebas sean más fáciles. Su motor de templates está basado en enlaces bidireccionales a los datos, esto significa que la vista (HTML) será actualizada automáticamente cuando cambie el modelo con el cual está enlazada.

Base de datos

La aplicación necesita almacenar los símbolos del LEL, los mensajes procesados, las reglas modeladas, etc., por lo que se necesita contar con un motor de base de datos, y hemos optado por MySQL, considerada la base de datos de código abierto más popular del mundo [13].

API Application

Este contenedor es central en la arquitectura de la aplicación, por que es la capa que lleva a caracterizarse como SOA. Permite abstraer todas las complejidades propias del dominio del sistema y brindársela como funciones de grano grueso para que sean utilizadas por otros contenedores o sistemas. En un sistema JEE como se propone, este contenedor estará actuando en el contexto de servidor de aplicación (application server) y con un conjunto de tecnologías, herramientas y frameworks. Para construirlo se utilizó el framework de aplicaciones Java llamado Spring, que se caracteriza por ser un contenedor liviano sobre el cual los servicios de negocio son implementados como objetos java simples (POJO, plain old java objects).

Sobre este contenedor haremos el próximo "zoom-in" dentro del diagrama de la arquitectura, expuesto en la próxima sección.

4.1.3. Nivel 3: Componentes del API Application

El principal contenedor de la herramienta está compuesto por los componentes: API Rest, Modelo de dominio, Motor de reglas, Manejador de eventos y la Capa de acceso. A continuación describiremos la función y las tecnologías empleadas en cada uno:

API Rest

En la actualidad existe una gran cantidad de proyectos o aplicaciones que disponen de una API REST, una nueva opción o estilo de uso de los Servicios Web (Web Services, WS), para la creación de servicios profesionales. REST es una interfaz entre sistemas que usa HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML, JSON, HTTP, etc. En los últimos años logró un gran impacto en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo más simple de utilizar y sobre todo por su eficiencia [21]. Para construirlo se optó por Spring, que define un conjunto de anotaciones

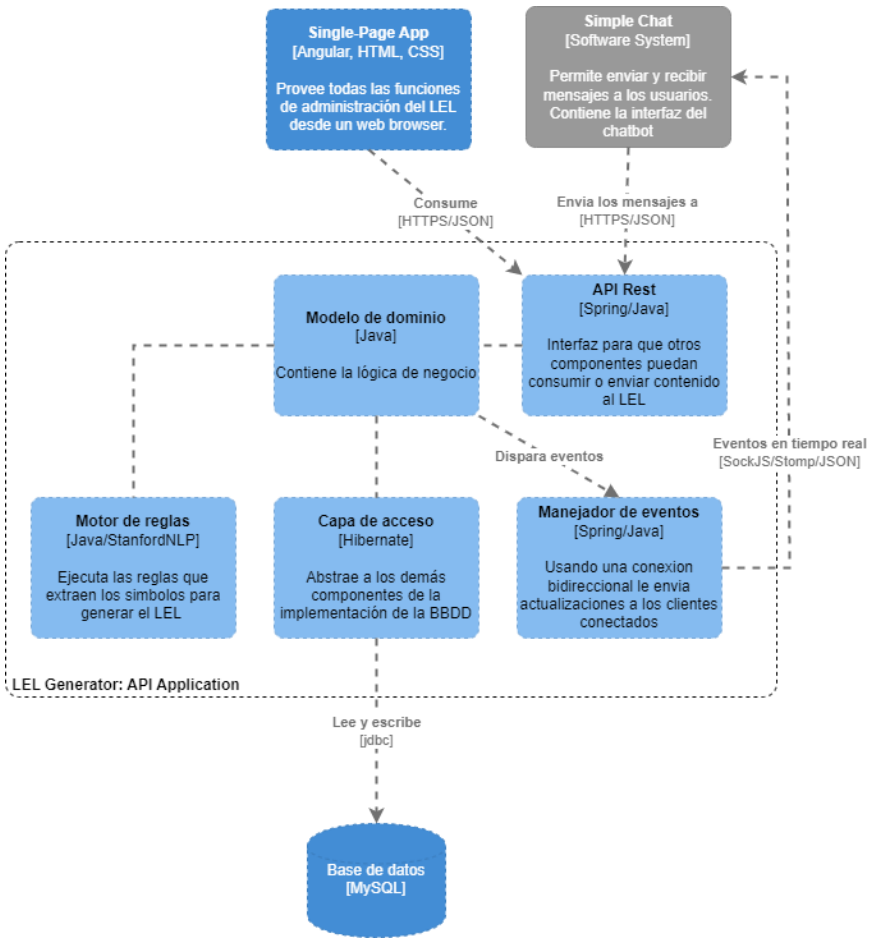

Figura 4.3: Diagrama de componentes del contenedor API Application

que permiten abstraerse de la implementación para concentrarse en el diseño de la API.

Modelo de dominio

Esta capa crea y contiene una red de objetos interconectados donde cada objeto encapsula la estructura y el comportamiento de una entidad de negocio. La lógica de negocios en un sentido amplio es un conjunto de métodos o procedimientos utilizados para administrar una función específica de negocio. El proceso de diseño orientado a objetos (Object Oriented Design, OOD), sobre el cual el sistema está basado, permite descomponer esa función

de negocios en un conjunto de elementos denominados objeto de negocios (Business Objects). La representación de un problema de negocios involucra varios objetos de negocio que interactúan para proveer la funcionalidad de negocios requerida. El conjunto de reglas específicas de negocio ayudan a identificar la estructura y el comportamiento de los objetos de negocio.

Motor de reglas

Dentro de la herramienta se define a una regla como un conjunto de pasos, donde cada uno esta compuesto de condiciones y acciones. Estas reglas son modeladas por los usuarios de la herramienta con el fin de capturar símbolos, nociones e impactos. El motor de reglas evalúa cada regla definida sobre los mensajes recibidos, emitiendo eventos al chat cada vez que el LEL se ve modificado. El motor de reglas emplea técnicas de NLP tales como POS Tagging, dependencias gramaticales, coreference resolution, entre otras. Para aplicar estas técnicas se opto por utilizar la librería CoreNLP [16], desarrollada por la universidad de Stanford para aplicaciones JAVA. Esta librería permite procesar un texto asignando a cada palabra un conjunto de anotaciones. Luego, el motor de reglas procesa tales anotaciones para evaluar las condiciones definidas en las reglas modeladas. El diseño del motor de reglas será descrito en la sección 4.3.

Manejador de eventos

La aplicación necesita que cuando un usuario envíe un mensaje en el chat, los demás usuarios conectados lo vean, esto podría pensarse como un evento que ocurre en la aplicación. Otros eventos podrían ser la identificación de un nuevo símbolo, o un nuevo impacto sobre un símbolo ya definido. Dado el esquema de trabajo elegido, habiendo aplicado en la vista un SPA, es coherente resolver este requerimiento con una tecnología que permita al servidor comunicarse con los clientes sin necesidad que previamente haya habido un request, como podría pensarse usando el protocolo HTTP y una estrategia de pooling. Por lo que se decidió emplear Websocket. WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor [32].

Capa de acceso

Para abstraernos del manejo de la persistencia, y el mapeo entre los datos de la base de datos y los objetos Java de nuestra aplicación, optamos por usar el ORM Hibernate. Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto, permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la POO. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

4.2. Herramientas utilizadas para el desarrollo

4.2.1. Editor de código Visual Studio Code

Visual Studio Code es un editor de código liviano pero potente que está disponible para Windows, macOS y Linux. Posee soporte incorporado para JavaScript, TypeScript y Node.js y además la posibilidad de incorporar extensiones para otros lenguajes (como C++ , C#, Python, PHP, Go) y runtimes (como .NET y Unity). Esta herramienta permite realizar diversas operaciones de desarrollo como depuración, ejecución de tareas y control de versiones. Su principal objetivo es proporcionar solo las herramientas que un desarrollador necesita para un ciclo rápido de desarrollo y depuración de código dejando los flujos de trabajo más complejos a los IDEs más completos. Es gratuito y de código abierto.

4.2.2. MySQL Workbench

MySQL Workbench es una herramienta visual unificada para arquitectos de bases de datos, desarrolladores y administradores de bases de datos (DBAs). Proporciona modelado de datos, desarrollo en SQL y herramientas de administración integral para la configuración de servidores, administración de usuarios, copia de seguridad, entre otros. Está disponible en

Windows, Linux y Mac OS X. Como características principales podemos mencionar:

- Diseño: permite a un DBA, desarrollador o arquitecto de datos diseñar, modelar, generar y administrar bases de datos visualmente. Incluye todo lo que un modelador de datos necesita para crear modelos de ER complejos, realizar ingeniería directa e inversa, y también ofrece funciones clave para realizar tareas difíciles de administración y documentación de cambios que normalmente requieren mucho tiempo y esfuerzo.

- Desarrollo: ofrece herramientas visuales para crear, ejecutar y optimizar consultas SQL. El Editor de SQL proporciona resaltado de sintaxis con color, autocompletado, reutilización de fragmentos de SQL e historial de ejecución de SQL. El Panel de conexiones de bases de datos permite a los desarrolladores administrar fácilmente las conexiones de bases de datos estándar, incluido MySQL Fabric.

- Administración: proporciona una consola visual para administrar fácilmente los entornos de MySQL y obtener una mejor visibilidad de las bases de datos. Los desarrolladores y DBA pueden usar las herramientas visuales para configurar servidores, administrar usuarios, realizar copias de seguridad y recuperación, inspeccionar datos de auditoría y ver el estado de la base de datos.

- Tablero de rendimiento: proporciona un conjunto de herramientas para mejorar el rendimiento de las aplicaciones MySQL. Los DBA pueden ver rápidamente los indicadores clave de rendimiento utilizando el Tablero de rendimiento. Los informes de rendimiento brindan una fácil identificación y acceso a puntos de conexión IO, declaraciones SQL de alto costo y más. Además, con un solo clic, los desarrolladores pueden ver dónde optimizar su consulta con el Visual Explain Plan.

- Tablero de rendimiento visual: proporciona un conjunto de herramientas para mejorar el rendimiento de las aplicaciones MySQL. Los DBA pueden ver rápidamente los indicadores clave de rendimiento utilizando el Tablero de rendimiento. Los informes de rendimiento brindan una fácil identificación y acceso a puntos de conexión IO, declaraciones SQL de alto costo, entre otros.

- Migración de base de datos: proporciona una solución completa y fácil de usar para migrar Microsoft SQL Server, Microsoft Access, Sybase

ASE, PostreSQL y otras tablas, objetos y datos de RDBMS a MySQL. Los desarrolladores y DBA pueden convertir rápida y fácilmente aplicaciones existentes para ejecutar en MySQL tanto en Windows como en otras plataformas. La migración también admite la migración de versiones anteriores de MySQL a las últimas versiones.

4.2.3. GitLab

GitLab es una plataforma de desarrollo colaborativo para alojar proyectos y crear software junto a millones de desarrolladores utilizando Git. Git es un software de control de versiones pensado para la eficiencia y confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos. Las principales características de GitLab son:

- Wiki para cada proyecto
- Página web para cada proyecto.
- Gráfico para ver cómo los desarrolladores trabajan en sus repositorios y bifurcaciones del proyecto.
- Funcionalidades como si se tratase de una red social, por ejemplo: seguidores.
- Herramienta para trabajo colaborativo entre programadores.

4.3. Diseño de reglas

Esta sección describe como se logró diseñar a las reglas para que los usuarios puedan identificar elementos del LEL. Primero se describirá como fueron diseñadas y luego el conjunto inicial de reglas previstas por la herramienta.

4.3.1. Definición de una regla

Se define una regla como un conjunto de pasos, donde cada uno tiene un conjunto de condiciones y acciones, que actúan sobre una sola palabra. Si las condiciones de todos los pasos de una regla evalúan verdadero, entonces

se ejecutarán sus acciones, impactando sobre el LEL capturado. Las condiciones de un paso son evaluadas sobre una palabra, según el orden en el que estén dentro de la regla. De esta forma, se podría modelar una regla de 2 pasos, donde el primero evalúe si la palabra es un sustantivo y un segundo paso que evalúe si la siguiente es un verbo (figura 4.4). De esa forma, por ejemplo, podríamos identificar un Sujeto.

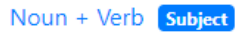

Figura 4.4: Ejemplo de regla con 2 pasos

La cantidad de pasos que componen a una regla son ilimitados, y la herramienta provee un conjunto variado de condiciones y acciones, para que los usuarios no estén limitados al modelar sus reglas. En la figura 4.5 se describen las clases que componen una regla.

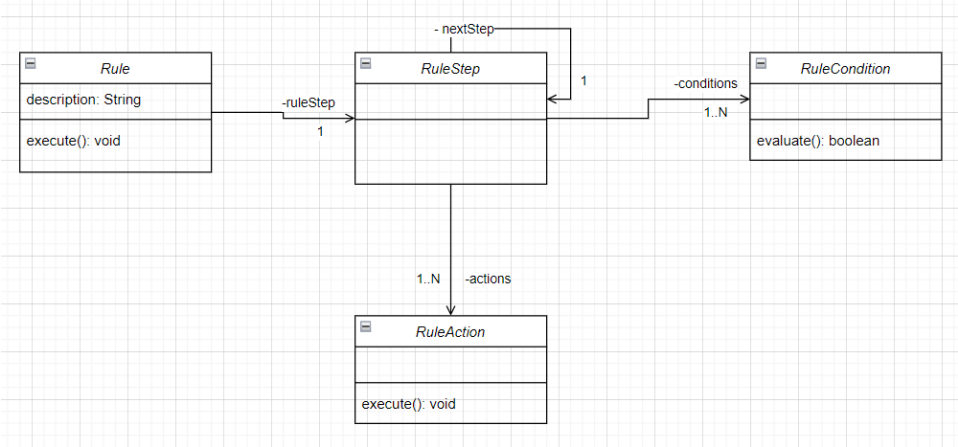

Figura 4.5: Diagrama de clases de una regla

Una vez modelada la regla, el sistema utiliza su estructura para describirla, facilitando su lectura. Así, cada paso es separado por el símbolo +, y es descrito en función de sus condiciones. Luego, sus acciones son representadas con tags, como se puede ver en las figuras 4.4 y 4.6.

A continuación se describirán los distintos tipos de condiciones y acciones disponibles en el sistema. En las figuras 4.7 y 4.10 se describen las clases

Figura 4.6: Ejemplo de regla con 4 pasos y múltiples acciones

que componen las condiciones y acciones respectivamente.

Condiciones

La herramienta provee 5 tipos de condiciones:

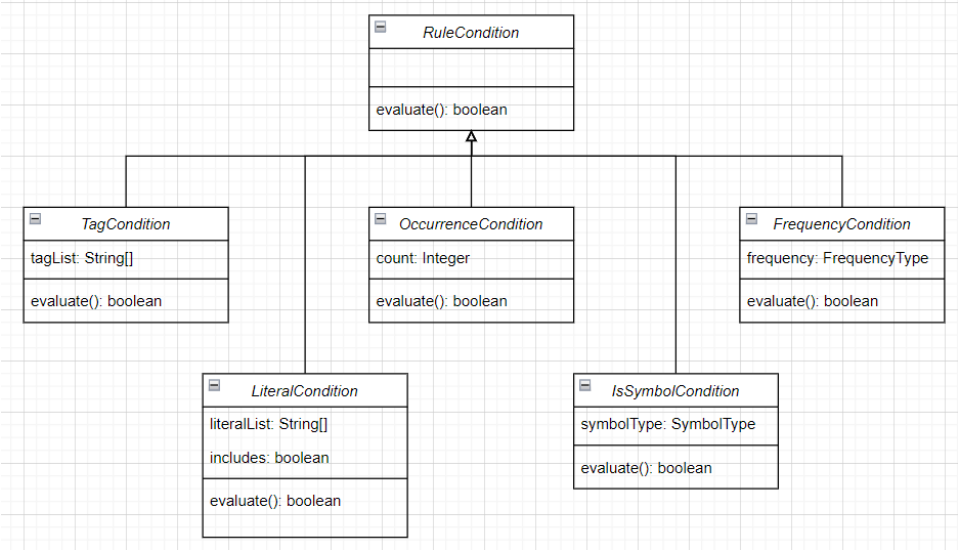

Figura 4.7: Diagrama de clases de las condiciones

TagCondition

Esta condición permite evaluar si la palabra cumple con un conjunto de PoS tags que especifica el usuario. Los tags utilizados son los definidos en Penn Treebank P.O.S Tags [33]. De esta manera los usuarios pueden modelar, por ejemplo, si la palabra es un sustantivo (tags NN, NNS). Todo paso de una regla debe tener una condición de Tag. Los tags a elegir son: adjetivos, adjetivos posesivos, sustantivos, pronombres personales, pronombres posesivos, adverbios y verbos. En el anexo A se encuentra el detalle de que

tags están disponibles. En la figura 4.8 se describe una regla de 3 pasos, que busca secuencias de sustantivos seguidos de verbos y luego otro sustantivo.

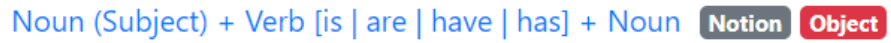

Figura 4.8: Ejemplo de regla con 3 pasos, usando sustantivos y verbo.

IsSymbolCondition

Esta condición permite evaluar si una palabra ya es un símbolo del LEL. Si un paso define esta condición, se verá reflejado el tipo de símbolo entre paréntesis, como se ve en la figura 4.8.

FrequencyCondition

Esta condición permite evaluar la frecuencia con la que fue mencionada una palabra. Los valores posibles son: "High" (5 menciones cada 100 palabras) y "Really high" (10 menciones cada 100 palabras).

LiteralCondition

Esta condición permite evaluar si la palabra esta dentro de un conjunto de palabras definidas por el usuario. Esta es la única condición que permite ser negada. Si un paso define esta condición, los literales definidos estarán listados entre corchetes. Como se ve en la figura 4.9, donde se evalúa que la palabra sea el verbo "have", "is" o "are".

Figura 4.9: Ejemplo de regla, cuyo segundo paso evalúa literales

OccurrenceCondition

Esta condición permite evaluar si una palabra fue mencionada más de N veces.

Acciones

La herramienta provee 8 tipos de acciones:

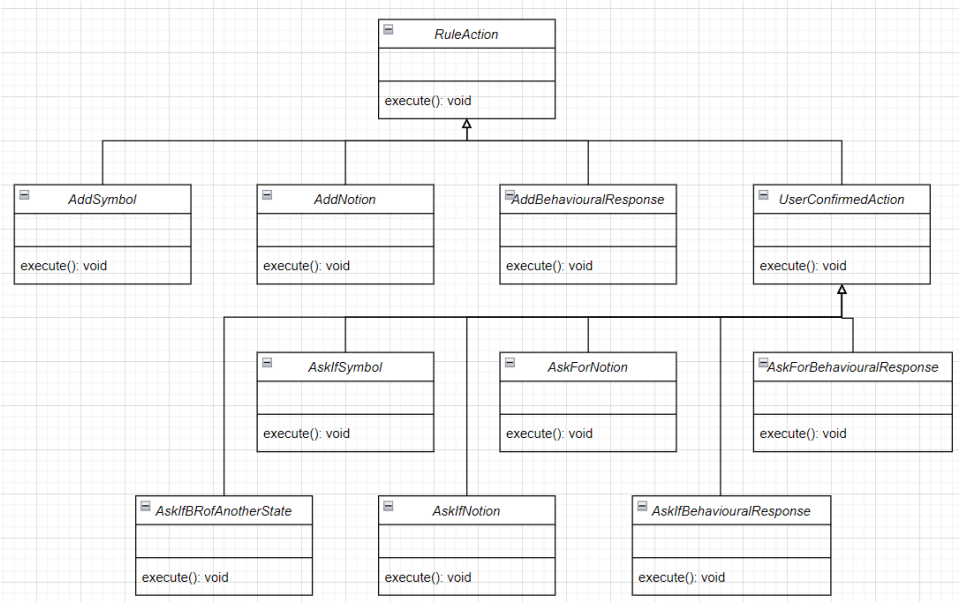

Figura 4.10: Diagrama de clases de las acciones

AddSymbolAction

Agrega la palabra como símbolo del LEL. Se debe especificar el tipo de símbolo: Sujeto, Objeto, Verbo o Estado. Si la regla posee este tipo de acción, cuando la misma es listada, se le agrega un tag con el nombre del simbolo. Por ejemplo, en la figura 4.6 se puede apreciar que la regla identifica verbos y objetos.

AddNotionAction

Agrega la oración como noción del símbolo al que representa la palabra evaluada. Esta acción requiere que el paso tenga la condición IsSymbolCondition o que agregue un nuevo símbolo con la acción AddSymbolAction. Agrega un tag gris con el texto "Notion" al final de la regla, como se ve en la figura 4.9.

AddBehaviouralResponseAction

Agrega la oración como impacto del símbolo al que representa la palabra evaluada. Esta acción requiere que el paso tenga la condición IsSymbolCondition o que agregue un nuevo símbolo con la acción AddSymbolAction. Agrega un tag amarillo con el texto "Behavioural response" al final de la

regla, como se ve en la figura 4.6.

AskIfSymbolAction
Hace uso del chatbot para preguntar a los usuarios del chat si una palabra es un símbolo del LEL. Requiere que el usuario ingrese el tipo de simbolo por el que preguntar.

AskIfNotionAction
Hace uso del chatbot para preguntar a los usuarios si la oración es la noción del símbolo que representa la palabra evaluada.

AskIfBehaviouralResponseAction
Mediante el chatbot se envía un mensaje al chat para preguntar a los usuarios si la oración es un impacto del símbolo que representa la palabra evaluada.

AskForNotionAction
Mediante el chatbot se envía un mensaje al chat para que los usuarios completen la noción de un símbolo.

AskForBehaviouralResponseAction
Mediante el chatbot se envía un mensaje al chat para que los usuarios completen los impactos de un símbolo.

4.3.2. Conjunto inicial de reglas

La herramienta provee un conjunto inicial de reglas, para poder procesar texto inmediatamente a modo de demostración. Luego, los usuarios podrán modificarlas o agregar nuevas, según lo crean conveniente. El objetivo de este conjunto de reglas es poder capturar un LEL, esto implica capturar sus símbolos, nociones e impactos. A continuación se describen las 14 reglas que componen este conjunto inicial:

Noun + Verb

Los sujetos son elementos activos que realizan acciones, por lo que una regla que los identifique sería un Noun seguido de un Verb.

Descripción: Identifica los símbolos de tipo sujeto.

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Acción: Agregar palabra como Sujeto.
- Paso 2

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Acción: Ninguna.

De esta forma, el análisis de los Pos tags para la oración 'The client can open an account' daría por resultado agregar la palabra 'client' como Sujeto dentro del LEL. El detalle de los pos tags puede verse en la tabla 4.1. Los PoS tags en gris son obviados por el motor de reglas, esto será analizado en el apartado 4.4.3.

Words	The	client	can	open	an	account
Pos tag	DT	NN	MD	VB	DT	NN

Cuadro 4.1: Análisis de la regla que captura sujetos

Noun (Subject) + Verb [is — are — have — has] + Noun

Esta regla permite capturar objetos y la noción de los sujetos. La noción de un sujeto se define como sus características o las condiciones que satisfacen, osea que describen relaciones "parte de" o "es un".

Descripción: A partir de un sujeto del LEL captura su noción, y el objeto relacionado.

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Condición: Es símbolo: Sujeto
  Acción: Agregar oración como noción del Sujeto.
- Paso 2

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Condición: Palabra literal: is, are, have, has

Acción: Ninguna.

- Paso 3

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Acción: Agregar la palabra como Objeto en el LEL.

Veamos como aplica esta regla en la oración: "Clients must have an account":

Words	Clients	must	have	an	account
Pos tag	NN	MD	VB	DT	NN

Cuadro 4.2: Análisis de la regla que captura objetos y la noción de los sujetos

Si "Client" ya es un símbolo del LEL, entonces la oración se agregará como noción de este. Además, la palabra "account" se agregará como Objeto.

Noun (Subject) + ![has — have — are — is]Verb + Noun

Los impactos de los sujetos son las acciones que estos realizan. De esta manera, podríamos preguntar por verbos que no sean los literales usados en la regla anterior, para obtener los impactos. Luego, también estaríamos detectando el Verbo y el Objeto de la acción como símbolos. Además, el impacto de los objetos son las acciones que se realizan sobre ellos, por lo que la misma oración es un impacto del objeto identificado. Como ya estamos identificando al verbo como símbolo, agregamos la acción para preguntarle a los usuarios por sus impactos.

Descripción: Regla que captura verbos, objetos, y los impactos del sujeto y del objeto identificado.

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Condición: Es símbolo: Sujeto
  Acción: Agregar oración como impacto del Sujeto.

- Paso 2

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Condición: Palabra literal NO es: is, are, have, has

Acción: Agregar palabra como Verbo.

Acción: Preguntar a los usuarios por los impactos del verbo identificado.

- Paso 3

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)

  Acción: Agregar la palabra como Objeto en el LEL.

  Acción: Agregar oración como impacto del Objeto identificado.

Veamos como aplica esta regla sobre la oración "The client can deposit money into his account":

Words	The	client	can	deposit	money	into	his	account
Pos tag	DT	NN	MD	VB	NN	IN	PRP	NN

Cuadro 4.3: Análisis de la regla que captura verbos, objetos e impactos

Al cumplirse las condiciones de la regla, la palabra "deposit" se agregaría al LEL como Verbo, y "money" como Objeto. Además, toda la oración se agregaría como impacto del sujeto "client" y el recién capturado objeto "money". Por último, se les preguntaría a los usuarios dentro del chat por los impactos del verbo "deposit".

Possessive pronoun (Subject) + Noun

Esta regla es capaz de capturar los atributos de los sujetos (noción), además de agregar a su atributo como un Objeto del LEL. Para poder resolver a quien hace referencia el pronombre posesivo se utiliza la técnica dependency parsing.

Descripción: Regla que captura la noción de un sujeto además de objetos del LEL.

Pasos:

- Paso 1

  Condición: PoS tag: Possesive pronoun (PRP)

  Condición: Es símbolo: Sujeto

  Acción: Agregar oración como noción del Sujeto.

- Paso 2

Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
Acción: Agregar palabra como Objeto.

Siguiendo el análisis de la oración "The client can deposit money into his account" como muestra la tabla 4.3, las condiciones cumplirían sobre "his account". De esta forma, la oración se agregaría como noción del sujeto "client", y se agregaría a "account" como objeto del LEL.

Noun

Esta regla evalúa si un sustantivo fue mencionado mas de 20 veces, en cuyo caso le pregunta a los usuarios si se trata de un Objeto.

Descripción: Regla que pregunta a los usuarios si una palabra es un Objeto del LEL

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Condición: Cantidad de ocurrencias mayor a: 20
  Acción: Preguntar al usuario si se trata de un Objeto del LEL.

Noun (Object) + Verb [have — is — are]

Para detectar las características u atributos de los objetos, se propone evaluar los verbos "have, is y are" cuando el sujeto sea un Objeto ya identificado del LEL.

Descripción: Regla que identifica las nociones de un Objeto.

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Condición: Es símbolo: Objeto
  Acción: Agregar oración como noción del Objeto.
- Paso 2

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Condición: Palabra literal es: have, is, are

Verb (Verb) + Noun + To

La noción de un Verbo es el objetivo que este persigue, es por esto que se propone utilizar el PoS tag "TO" (para) luego del Verbo.

Descripción: Regla que identifica las nociones de un Verbo.

Pasos:

- Paso 1

  Condición: PoS tag: Verb (VB, VBP, VBZ)

  Condición: Es símbolo: Verbo

  Acción: Agregar oración como noción del Objeto.
- Paso 2

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
- Paso 3

  Condición: PoS tag: TO

Veamos como esta regla evalúa sobre la oración "He opened an account to start trading goods":

Words	He	opened	an	account	to	start	trading	goods
Pos tag	PRP	VBD	DT	NN	TO	VB	VBG	NNS

Cuadro 4.4: Análisis de la regla que captura la noción de un Verbo.

Si el verbo "open" es un símbolo del LEL entonces la oración sería agregada como su noción.

Verb (Verb) + To

Al igual que la regla anteriormente definida, con esta propuesta también puede detectarse la noción de un símbolo Verbo.

Descripción: Otra regla para detectar las nociones de un Verbo.

Pasos:

- Paso 1

  Condición: PoS tag: Verb (VB, VBP, VBZ)

Condición: Es símbolo: Verbo

Acción: Agregar oración como noción del Objeto.

- Paso 2

  Condición: PoS tag: TO

Adjective + Noun (Subject) / Adjective + Noun (Object)

Un estado es una situación en la que puede estar un Sujeto u Objeto. De esta manera se propone identificarlos utilizando los adjetivos en la oración, seguidos de un Sujeto u Objeto ya identificado del LEL. Luego, su noción es la situación que representa y sus impactos las acciones que deben ser realizadas para cambiar a otro estado. Por la dificultad que supone identificar tanto su noción como impactos, se decide que esta regla además de identificar un Estado, pregunte a los usuarios por su noción e impactos mediante el chatbot.

Descripción: Regla para identificar Estados.

Pasos:

- Paso 1

  Condición: PoS tag: Adjective (JJ)

  Acción: Agregar palabra como Estado.

  Acción: Preguntar a los usuarios del chat por su noción e impactos.

- Paso 2

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)

  Condición: Es símbolo: Sujeto u Objeto.

Veamos como esta regla aplicaría sobre la siguiente oración:

Words	Clients	can	not	withdraw	money	from	a	closed	account
Pos tag	NNS	MD	RB	VB	NN	IN	DT	JJ	NN

Cuadro 4.5: Análisis de la regla que captura estados.

Si la palabra “Account” es un Objeto del LEL, entonces la palabra “closed” será identificada como Estado del LEL. Además, el chatbot preguntará a los usuarios del chat por la noción e impactos del estado recién identificado.

Noun (Subject) + Verb ![is — are — have — has] + Adjective + Noun

Esta regla propuesta se obtiene de tomar la regla **Noun (Subject) + Verb ![is — are — have — has] + Noun** definida anteriormente, pero agregando un paso adicional, para evaluar los adjetivos que podrían aplicar sobre el ultimo sustantivo. De esta forma, su definición es:

Descripción: Regla que captura verbos, objetos, el estado del objeto, y los impactos del sujeto y del objeto identificado.

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Condición: Es símbolo: Sujeto
  Acción: Agregar oración como impacto del Sujeto.
- Paso 2

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Condición: Palabra literal NO es: is, are, have, has
  Acción: Agregar palabra como Verbo.
- Paso 3

  Condición: PoS tag: Adjective (JJ, JJR, JJS)
  Acción: Agregar la palabra como Estado en el LEL.
- Paso 4

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Acción: Agregar la palabra como Objeto en el LEL.
  Acción: Agregar oración como impacto del Objeto identificado.

Noun (Subject) + Verb + Personal pronoun + Noun

Con esta regla proponemos capturar Objetos, Verbos, los impactos de los Sujetos y Objetos, considerando los pronombres personales de los Sujetos del LEL.

Descripción: Regla que captura verbos, objetos, e impactos, considerando pronombres personales.

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Condición: Es símbolo: Sujeto
  Acción: Agregar oración como impacto del Sujeto.
- Paso 2

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Acción: Agregar palabra como Verbo.
- Paso 3

  Condición: PoS tag: Personal pronoun (PRP)
- Paso 4

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Acción: Agregar la palabra como Objeto en el LEL.
  Acción: Agregar oración como impacto del Objeto identificado.

Noun (Subject) + Verb + To + Verb + Noun

Con esta regla proponemos detectar Verbos y Objetos en oraciones complejas.

Descripción: Regla que captura verbos y objetos.

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Condición: Es símbolo: Sujeto
  Acción: Agregar oración como impacto del Sujeto.
- Paso 2

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Acción: Agregar palabra como Verbo.
  Acción: Agregar oración como noción del Verbo.
- Paso 3

  Condición: PoS tag: To (TO)
- Paso 4

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Acción: Agregar palabra como Verbo.

- Paso 5

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Acción: Agregar palabra como Objeto.
  Acción: Agregar oración como impacto del Objeto.

Verb (Verb) + Adjective + Noun + To

Esta regla toma como base la regla ya definida **Verb (Verb) + Noun + To** pero añade un paso para incluir también a las oraciones que antes del sustantivo tengan un adjetivo.

Descripción: Regla que captura Objetos y la noción de los Verbos.
Pasos:

- Paso 1

  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Condición: Es símbolo: Verbo
  Acción: Agregar oración como noción del Verbo.
- Paso 2

  Condición: PoS tag: Adjective (JJ, JJS, JJR)
- Paso 3

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
  Acción: Agregar palabra como Objeto.
  Acción: Agregar oración como impacto del Objeto.
- Paso 4

  Condición: PoS tag: To (TO)

4.4. Ejecución

Desde que se ingresa un mensaje hasta que se realizan cambios en el LEL capturado ocurren muchas cosas, que vamos a bajar a detalle. Los textos de los mensajes no son modificados de ninguna forma, se los procesa tal cual fueron escritos por los usuarios. Luego, se aplican técnicas de PLN para anotar las palabras con información relevante que luego serán procesadas por las reglas. Se optó por utilizar la librería CoreNLP [16] ya que esta disponible para entornos Java y su documentación es extensa. Además, se mantiene un registro de las ocurrencias de cada palabra, de esta forma se logra utilizar

condiciones en base a la frecuencia con que se menciona una palabra. Al recibir un mensaje, puede que se este haciendo mención a elementos de mensajes anteriores, es por esto que la herramienta resuelve las referencias que existan antes de procesar las reglas modeladas. En la figura 4.11 se visualiza el flujo de ejecución completo desde que ingresa un mensaje hasta que se impacta en el LEL.

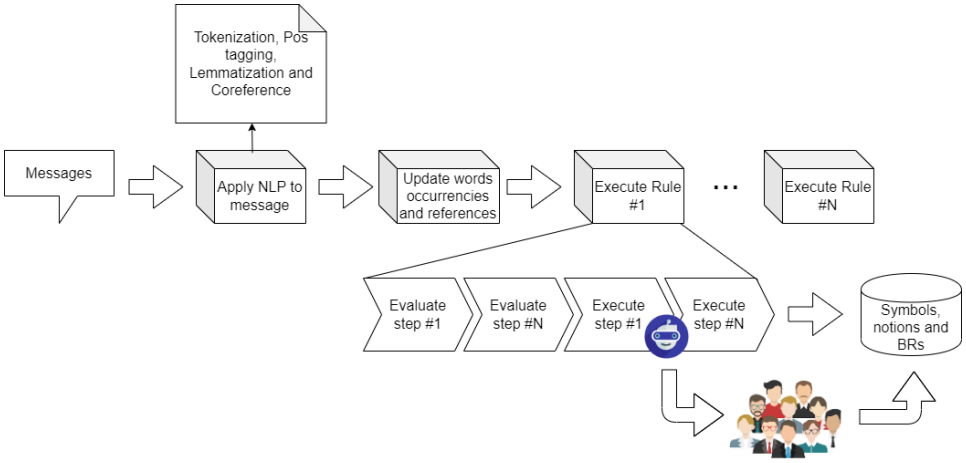

Figura 4.11: Diagrama de la ejecución de LELGenerator

La ejecución del motor de reglas comienza en el método "processMessage" de la clase RuleEngine, este es el punto de partida. El código de dicho método puede verse en la figura 4.12.

Cada uno de los pasos en la ejecución del motor de reglas será detallado a continuación.

4.4.1. Configuración de la librería CoreNLP

El paso inicial es configurar la librería CoreNLP para indicarle que procesamiento queremos que realice sobre el texto a analizar. Esto se configura dentro de un bloque estático de la clase RuleEngine, utilizando la propiedad "annotators", como se ve en la figura 4.13.

A continuación se describe el por qué de cada técnica empleada:

Tokenize

Es la técnica que permite dividir el texto en tokens, unidades indivisibles de texto. Es necesaria por las demás técnicas para que funcionen.

```
public void processMessage(Message m) {
    if(m.getText() != null && !"".equals(m.getText().trim())) {
        //first we generate the sentences for that message
        List<Sentence> sentences = this.generateSentences(m);

        if(!sentences.isEmpty()) {
            //then, we need to run all rules on each sentence.
            List<Rule> xRules = ruleController.getRules();

            for(Sentence sentence : sentences) {
                for(Rule r : xRules) {
                    r.execute(m, sentence);
                }
            }
        }
    }
}
```

Figura 4.12: Método que procesa un mensaje recibido del chat

Ssplit

Es la abreviación de "Sentence Splitting", permite dividir el texto en oraciones. Esto es clave, ya que las reglas se evalúan sobre cada oración.

Pos

Es la abreviación de "Part of speech", anota en cada token su categoría gramatical. Esto es necesario para luego poder evaluar las condiciones de las reglas.

Lemma

Indica que se debe realizar el proceso de lematización, anotando en cada token su raíz. Esto es importante, ya que la herramienta lo utilizará para evitar almacenar un mismo símbolo escrito de distintas maneras.

Depparse

Indica que debe realizarse el procesamiento de las dependencias gramaticales [40] entre las palabras del texto. La librería representa estas relaciones utilizando el Esquema de Universal Dependencies [12].

```java
@Service
public class RuleEngine {
    @Autowired
    private SentenceController sentenceController;
    @Autowired
    private RuleController ruleController;
    @Autowired
    private WordOccurrencesEngine wordOccurrencesEngine;
    private static StanfordCoreNLP pipeline = null;

    static {
        Properties props = new Properties();
        props.setProperty("annotators", "tokenize, ssplit, pos, lemma, coref, depparse");
        pipeline = new StanfordCoreNLP(props);
    }
```

Figura 4.13: Configuración de la librería CoreNLP

Coref

Es la abreviación de "Coreference resolution" [39], indica que debe realizarse la búsqueda de menciones de una misma entidad en el texto, por ejemplo "Juan Altamirano" y "él". Esto es importante, ya que en un mensaje se puede estar haciendo referencia a entidades mencionadas en mensajes anteriores.

4.4.2. Dividir mensaje en oraciones y anotarlo

Una vez que la librería CoreNLP esta configurada, el sistema esta listo para procesar mensajes. Al ingresar uno, el primer paso es analizarlo utilizando la librería. Esto permite partir el texto en oraciones, donde cada una de sus palabras tiene un conjunto de anotaciones con meta-data necesarias para el posterior análisis de las reglas. Luego, cada oración es procesada para actualizar la ocurrencia en que aparecen las palabras de la misma y resolver las referencias que existiesen, esto será descrito en el próximo apartado. En la figura 4.14 se visualiza el método encargado de esta tarea.

4.4.3. Categorías gramaticales de interés

Al etiquetar cada palabra con su categoría gramatical el sistema logra identificar el rol de la palabra dentro de la oración. El sistema permite a los usuarios utilizar estas categorías dentro de las condiciones a cumplir por las reglas. Dentro del idioma Ingles existen muchos roles que puede tomar una palabra, para simplificar el desarrollo de la herramienta se optó por procesar un subconjunto de estas posibles categorías gramaticales. En el anexo A se detallan cuales categorías son de interés para la herramienta y cuales no.

```
private List<Sentence> generateSentences(Message m) {
    Annotation annotation = new Annotation(m.getText());
    pipeline.annotate(annotation);
    List<CoreMap> annotatedSentences = annotation.get(CoreAnnotations.SentencesAnnotation.class);
    List<Sentence> result = new ArrayList<Sentence>();
    for(CoreMap annotatedSentence : annotatedSentences) {
        result.add(sentenceController.createSentence(
                new Sentence(annotatedSentence.get(CoreAnnotations.TextAnnotation.class),
                        m, annotatedSentence)));
        wordOccurrencesEngine.processSentence(annotatedSentence);
    }
    processReferences(result);
    return result;
}
```

Figura 4.14: Método que procesa el texto para generar oraciones con anotaciones gramaticales

4.4.4. Procesar ocurrencias y referencias

La herramienta permite definir condiciones que contemplan la ocurrencia de las palabras. Para resolver este tipo de condiciones se aplica un procesamiento adicional de cada oración del mensaje recibido. Este procesamiento consiste en tomar cada palabra y en caso de ser candidata a ser analizada, en función de si es una stopword y si su rol gramatical dentro de la oración es de interés para el análisis, se aumenta un contador que persiste en base de datos. Luego, se analiza cada oración en busca de referencias a entidades en oraciones anteriores, esto se realiza utilizando las anotaciones generadas por la técnica de Coreference Resolution. En caso de que la regla evalúe una palabra que hace referencia a una entidad de otra oración, y se decida agregarla como simbolo, en realidad se agrega la entidad referida. De esta manera, de recibirse los mensajes “Administrators are an important part of the system” seguido de “They manage accounts.” Al procesar el segundo mensaje la palabra “They” haría referencia a “Administrators”, agregándolos como sujetos, por cumplirse la regla inicial noun+verb+noun. La figura 4.15 muestra como la librería CoreNLP establece las referencias entre las palabras en dicho caso.

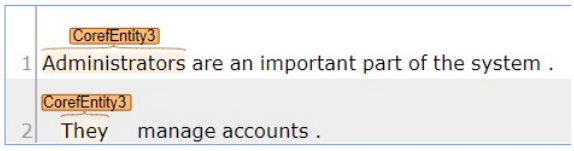

Figura 4.15: Ejemplo de oraciones anotadas con la técnica de Coreference Resolution

4.4.5. Algoritmo de ejecución de reglas

En este punto, las oraciones del mensaje recibido ya están anotadas con toda la información necesaria para evaluar las reglas modeladas. Este proceso es recursivo ya que se deben evaluar los pasos de la regla en el orden correcto, hasta que todas las condiciones de cada paso evalúen verdadero. La figura 4.16 muestra la codificación de este algoritmo dentro de la clase RuleStep.

```
public int evaluate(int pIndex, Sentence sentence, SearchPolicy policy) {
    this.index = pIndex;
    //System.out.println("Evaluando step: "+this+
    //" sentence: "+sentence+" index: "+this.index);
    if(this.index == 0) {
        this.firstStep = true;
    }
    CoreMap currentAnnotatedWord = getNextWord(sentence);
    if(currentAnnotatedWord != null) {
        if(policy == SearchPolicy.STRICT) {
            if(!checkConditions(currentAnnotatedWord, sentence)) {
                currentAnnotatedWord = null;
            }
        } else {
            while(currentAnnotatedWord != null &&
                    !checkConditions(currentAnnotatedWord, sentence)) {
                this.index++;
                currentAnnotatedWord = getNextWord(sentence);
            }
        }
        if(currentAnnotatedWord != null) {
            //jackpot!
            int result;
            if(nextStep != null) {
                result = nextStep.evaluate(this.index + 1, sentence);
                if(result > -1) {
                    executeActions(currentAnnotatedWord, sentence);
                }
            } else {
                executeActions(currentAnnotatedWord, sentence);
                result = this.index + 1;
            }
            if(this.firstStep) {
                if(result == -1) {
                    result = this.index + 1;
                }
                //ok, we just executed the first step actions.
                //Now, we have to keep evaluating this rule in the rest of the sentence.
                this.evaluate(result, sentence, SearchPolicy.LOOSE);
            }
            return result;
        }
    }
    return -1;
}
```

Figura 4.16: Algoritmo recursivo que evalúa un paso de una regla

La primer ejecución de este algoritmo se realiza con la configuración del primer paso de la regla a evaluar, y la palabra con índice cero dentro de la oración. Luego, se evalúa si la palabra en cuestión cumple las condiciones

del paso, de no ser así, se itera sobre las demás palabras hasta encontrar alguna que cumpla las condiciones. Si el primer paso de la regla encuentra una palabra que cumpla sus condiciones se debe evaluar si existe un siguiente paso modelado en la regla o si ya estábamos evaluando el ultimo. Si existe un siguiente paso, el método se vuelve a llamar a si mismo pero con la configuración de dicho paso. Si no existía siguiente paso significa que todos los pasos de la regla evaluaron verdadero, por lo que las acciones son ejecutadas. El método retorna -1 si el paso actual no cumplió las condiciones, de esta forma al retornar el control al paso anterior, este puede saber si el siguiente evaluó verdadero, y de esta forma ejecutar sus acciones.

4.4.6. Evaluar las condiciones de una regla

La clase RuleStep que representa un paso de una regla, contiene una lista de condiciones modeladas. Estas condiciones se representan con la clase RuleCondition. El algoritmo que procesa las condiciones de un paso se muestra en la figura 4.17. Se decidió utilizar polimorfismo para resolver cada tipo de condición particular, esto puede verse en el diagrama de clase de las condiciones 4.7.

```
130    private boolean checkConditions(CoreMap word, Sentence s) {
131        for(RuleCondition condition : conditions) {
132            if(!condition.evaluate(word, s, this)) {
133                return false;
134            }
135        }
136        return true;
137    }
```

Figura 4.17: Algoritmo que evalúa las condiciones de un paso de una regla

4.4.7. Ejecutar las acciones de una regla

Si las condiciones de todos los pasos de una regla se cumplen, entonces el sistema debe ejecutar las acciones modeladas en cada paso. Esto se realiza en el método executeActions de la clase RuleStep, dicho código puede verse en la figura 4.18. Al igual que las condiciones, cada tipo de acción esta representada por una clase específica, esto puede verse en la figura 4.10.

Dentro de las acciones que el sistema permite modelar, se debe destacar la acción que permite añadir una palabra como símbolo. Dependiendo el tipo de símbolo a agregar, se realiza un procesamiento adicional:

```
139    private void executeActions(CoreMap annotatedWord, Sentence sentence) {
140        for(RuleAction action : actions) {
141            action.execute(annotatedWord, sentence, this);
142        }
143    }
```

Figura 4.18: Algoritmo que ejecuta las acciones de un paso de una regla

- Si el símbolo a agregar es un Verbo, Sujeto u Objeto, se toma el lemma de la palabra. Esto evita mencionar a los símbolos en formas flexionadas.
- Si el símbolo a agregar es un Verbo, se agrega el objeto directo involucrado. Esto se realiza analizando el grafo de dependencias gramaticales obtenidas por la librería CoreNLP.
- Si el símbolo a agregar es un Sujeto u Objeto, se evalúa si la palabra es compuesta. Nuevamente se utilizan dependencias gramaticales para resolverlo.
- Si la palabra a agregar como símbolo hace referencia a otra entidad, se debe agregar como símbolo a la entidad referenciada. De esta manera, se evita agregar como símbolo a palabras como “He”.

Por último, antes de crear un símbolo, el sistema verifica que no exista en una lista de falsos positivos llamada Blacklist, esto se desarrollará a continuación.

4.4.8. Blacklist

Las reglas modeladas no son infalibles, se pueden detectar símbolos que en realidad no lo sean. Para que el sistema no vuelva a cometer los mismos errores, se desarrollo una blacklist, que es una lista de símbolos que el usuario indico que fueron falsos positivos. Los falsos positivos se agregan a la blacklist en las siguientes situaciones:

- Cuando los usuarios eliminan un símbolo.
- Cuando los usuarios descartan la propuesta de agregar un nuevo símbolo detectado por una regla. Esta situación se origina por la ejecución de la acción AskIfSymbolAction.
- Cuando los usuarios agregan explícitamente un nuevo símbolo a la blacklist, utilizando la UI de la herramienta.

Capítulo 5

Caso de estudio

Este capítulo abordará un caso de estudio para demostrar la ejecución tanto de las reglas iniciales propuestas como del motor de reglas. Esto es importante destacarlo, ya que la herramienta propuesta es capaz de procesar cualquier regla modelada por los analistas, de esta forma lo importante es comprobar que lo modelado fue correctamente ejecutado sobre los datos de prueba. De esta forma se aclara que el objetivo no es demostrar que las reglas iniciales conforman el universo completo de reglas a modelar para generar un LEL sin errores. Para realizarlo se modelaron las reglas propuestas como iniciales en esta tesina, y se simuló un chat de ejemplo donde un conjunto de analistas interactúan con el dueño de una empresa que requiere de software para resolver situaciones particulares de su negocio. Este capítulo está compuesto por las siguientes secciones: la sección 5.1 presenta el chat de ejemplo a procesar. Luego, la sección 5.2 presenta los resultados obtenidos. A continuación en la sección 5.3 se realiza el análisis de éstos. Al final, la sección 5.4 presenta las conclusiones del estudio realizado.

5.1. Descripción del Chat

Los datos de entrada utilizados representan el chat entre analistas y el dueño de una empresa. Él mismo se da en el marco de la elicitación de requerimientos de un nuevo sistema solicitado por el cliente. A continuación se presentan los participantes del chat y los mensajes.

5.1.1. Participantes

Nelson Castañeda - @nelson (General Manager at Club Lawn Tennis)
Edgardo Crocco - @ecrocco (Gerente de Área de Procesos de Negocio en Optaris S.A)
Juan Altamirano - @juancho (Analista Jr en Optaris S.A)

5.1.2. Mensajes

@ecrocco: Nelson, are you ready to start our meeting?
@nelson: Indeed, i'm ready
@ecrocco: great, as you know, we have an easy to use tool to automate process workflows, i'm sure we can help you increase your business productivity.
@ecrocco: so, tell us, what's your business about?
@nelson: i'm the manager of Lawn Tennis, one of the biggest tennis clubs in Lima.
@nelson: we have more than 300 members, and 20 tennis courts.
@nelson: but today we have the club almost closed, we are preparing for the government to let us work, the coronavirus protocol is going to be approved soon so i think it's a good opportunity to automate all the paperwork we have today
@ecrocco: yes, it's the perfect time!!
@ecrocco: so, let's begin with the basics, tell us what operations do you wish to automate?
@nelson: well.. i would like to have a beautiful webpage to start...
@nelson: where our members can login to pay fees, make reservations, and so on.
@juancho: nelson, could u tell us more about the club fees?
@nelson: every member pay a monthly fee to keep being an active member.
@nelson: only active members can reserve courts.
@nelson: we keep an excel sheet for every member...
@juancho: we will change that pretty quick :)
@nelson: i hope so!!, we have sooo many papers going around.. its annoying.. we try our best to keep track of unpaid fees, but its really hard
@ecrocco: well.. we will design a payments workflow to automate member's fees.
@ecrocco: how do the reservations work?
@nelson: we have another excel sheet, one per day..

@nelson: members make reservations by calling the club's phone number.
@nelson: then, our employees search for a free court for that day, and we mark it as reserved.
@nelson: all reserved courts have to be confirmed by the member 2 hours before the match, or we cancel the reservation.
@nelson: of course our members can cancel their reservations anytime

5.2. Resultado obtenido

A continuación se describe el LEL obtenido.

5.2.1. Objetos

- Fee
  - Impactos
    - Where our members can login to pay fees, make reservations, and so on.
- Tennis court
  - Noción
    - All reserved courts have to be confirmed by the member 2 hours before the match, or we cancel the reservation.
  - Impactos
    - Only active members can reserve courts.
    - Then, our employees search for a free court for that day, and we mark it as reserved.
- Reservation
  - Impactos
    - Members make reservations by calling the club's phone number.
    - Of course our members can cancel their reservations anytime.

5.2.2. Sujetos

- Coronavirus protocol
- Operation
- Member
  - Impactos
    - Members make reservations by calling the club's phone number.
    - We will design a payments workflow to automate member's fees.
    - Only active members can reserve tennis courts.
    - Every member pay a monthly fee to keep being an active member.
    - Where our members can login to pay fees, make reservations, and so on.
    - Of course our members can cancel their reservations anytime
- Payment workflow
  - Impactos
    - We will design a payments workflow to automate member's fees.
- Employee
  - Impactos
    - then, our employees search for a free court for that day, and we mark it as reserved.

5.2.3. Verbos

- Pay fee
  - Noción
    - Every member pay a monthly fee to keep being an active member.
- Login

- Noción
  - Where our members can login to pay fees, make reservations, and so on.

- Reserve tennis court
- Automate fee
- Make reservation
- Search
- Cancel reservation

5.2.4. Estados

- Active member
- Monthly fee
- Unpaid fee
- Free court

5.3. Análisis de los resultados

De los resultados obtenidos, podemos afirmar que:

- Todos los verbos, estados y objetos detectados serían correctos.
- 3 sujetos detectados en realidad serían objetos. En la oración "the coronavirus protocol is going to be approved soon so..." la regla "Noun + Verb" para identificar Sujetos se cumplió, por lo que marco a Coronavirus protocol como sujeto. Lo mismo ocurrió al analizar la oración "what operations do you wish to automate?", identificando como Sujeto a "Operations". En la oración "we will design a payments workflow to automate member's fees" la palabra "to" fue categorizada con el PoS tag "IN" por lo que el motor la ignoró en el análisis, haciendo que la regla "Noun + Verb" aplique, agregando a "Payments workflow" como sujeto.
- Las nociones detectadas para los verbos serían correctas.
- De 6 impactos detectados para el sujeto Member, solo 1 es incorrecto.

- Los impactos detectados para los objetos serían correctos.
- La noción detectada para el objeto Tennis Court es incorrecta.
- Al momento de detectar los estados, el Bot le preguntó a los usuarios por la situación que representa (noción del estado).
- El Bot le preguntó a los usuarios por las acciones que pueden hacer cambiar de estado, por cada estado detectado (impactos de los estados).
- El Bot preguntó en el chat por los pasos necesarios para realizar cada verbo detectado (impactos de los verbos).

Algunas de las preguntas hechas por el Bot del sistema pueden verse en la figura 5.1.

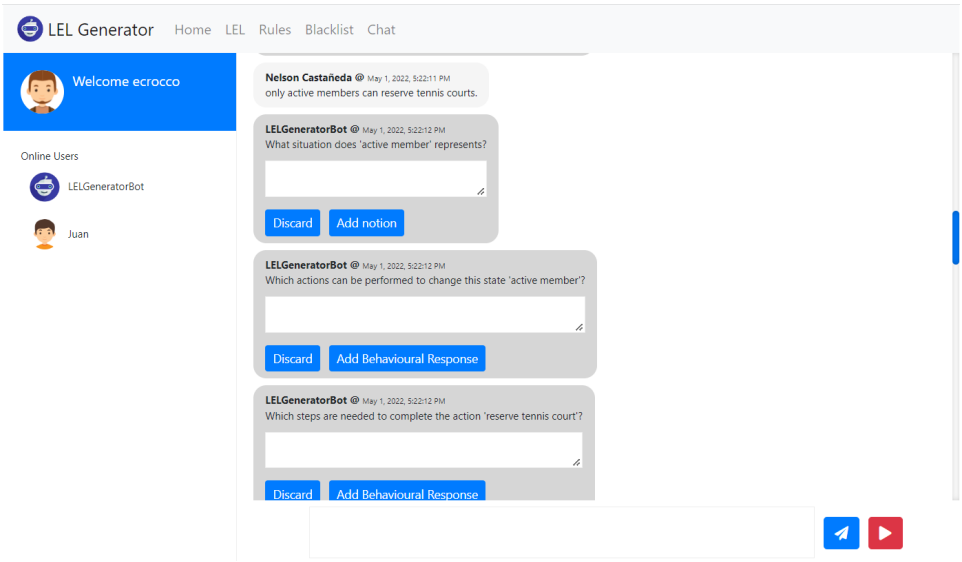

Figura 5.1: Preguntas hechas por el Bot al procesar el chat de ejemplo

5.4. Conclusiones

El motor de reglas ejecutó correctamente el set de reglas propuesto sobre el chat de ejemplo, identificando un conjunto de símbolos. Existieron símbolos correctamente detectados pero mal categorizados, producto de la regla "Noun + Verb" que es una forma general de identificar Sujetos.

Otra estrategia para detectar Sujetos podría haber sido utilizar la misma regla, pero preguntando a los usuarios si se trata de un Sujeto antes de agregarlo, utilizando la acción "AskIfSymbolAction". Otra estrategia podría haber sido utilizando la frecuencia con que aparecen las palabras, y así evitar identificar palabras como "Coronavirus protocol" que fueron mencionadas una sola vez en todo el chat.

No obstante esto, se aprecia la falta de análisis de las relaciones entre las palabras analizadas. Sería una mejora importante agregar a las condiciones la posibilidad de preguntar por el metadata generado por la técnica de PLN "Dependencias gramaticales".

Capítulo 6

Evaluación de usabilidad

6.1. Introducción

Se decidió llevar a cabo una prueba de usabilidad de la herramienta LEL-Generator. Para ello se utilizó la Escala de Usabilidad del Sistema, conocido como SUS, por sus siglas en inglés. La escala de usabilidad del sistema proporciona una herramienta confiable y rápida para medir la usabilidad. Fue creada originalmente por John Brooke [6]. Permite evaluar una amplia variedad de productos y servicios, incluidos hardware, software, dispositivos móviles, sitios web y aplicaciones.

6.2. Escala de Usabilidad del Sistema

La escala de usabilidad del sistema consiste en un cuestionario de 10 items con cinco opciones de respuesta para los encuestados desde "fuertemente en desacuerdo" a "fuertemente de acuerdo".

Los items son los siguientes:

1. Pienso que me gustaría usar este producto frecuentemente.
2. Encontré el producto innecesariamente complejo.
3. Pienso que el producto es fácil de usar.
4. Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.
5. Encontré que varias funciones en el producto estaban muy bien integradas.

6. Pienso que había mucha inconsistencia en el producto.
7. Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.
8. Encontré el producto muy incómodo de usar.
9. Me sentí muy seguro usando el producto.
10. Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.

Cada uno de los encuestados debe responder cada uno de los items. A continuación, se muestra el formato de respuesta del SUS:

Fuertemente en desacuerdo 1	2	3	4	Fuertemente de acuerdo 5

Figura 6.1: Formato de respuesta del SUS

Por cada ítem de los mostrados anteriormente, el encuestado marca con una cruz uno de los casilleros indicando su respuesta. Finalmente se calcula la puntuación del SUS de la siguiente manera:

- Para los items impares: a la respuesta del encuestado se le resta 1 (uno).
- Para los items pares: a 5 (cinco) se le resta la respuesta del encuestado. Esto escala todos los valores de 0 a 4 (siendo cuatro la respuesta más positiva)
- Se suma las respuestas de cada usuario y se multiplica ese total por 2,5. Esto convierte el rango de valores posibles de 0 a 100.

El rango recomendado para la medición de usabilidad según las clasificaciones de la escala de usabilidad del sistema [30], es el siguiente:

- 0 - 64: No aceptable
- 65 - 84: Aceptable
- 85 - 100: Excelente

El principal valor del SUS es que proporciona una puntuación de referencia única para la visión de los participantes de la usabilidad del producto.

6.3. Preparación

Se definieron reglas para modelar, y algunos símbolos de un LEL para ser utilizados en la prueba. La idea fue que los participantes tuvieran una guía de qué cargar en la aplicación manteniendo una coherencia y un sentido en la información cargada en la aplicación. En el Anexo B se encuentran las reglas y los símbolos del LEL a cargar.

Por último, se enviaron mails con el cuestionario para que cada participante pueda responder en forma individual. El cuestionario enviado puede verse en la figura 6.2.

6.4. Desarrollo

Se reunió a 6 personas, todas del ámbito de tecnología desde desarrolladores a líderes de proyecto, para que carguen en la aplicación LELGenerator los datos definidos en el apartado anterior.

En primer lugar, se realizo una introducción de que es el Léxico Extendido del Lenguaje y se explicó brevemente cómo se definen los símbolos y las reglas. También se explicaron las funcionalidades que provee la herramienta además de la carga de símbolos de un LEL.

Luego, se entregaron las definiciones de los distintos símbolos y reglas que cada participante debía cargar en la aplicación. Finalmente, se realizaron dos rondas de trabajo, en la primera, tomando turnos de a uno. En la segunda ronda, se permitió que cada participante en forma autónoma, es decir sin indicaciones, revise y utilice el resto de las funcionalidades disponibles.

Aclaraciones:

- Por favor tilde el casillero que refleje su respuesta inmediata a cada ítem.
- No piense mucho sobre cada ítem.
- Asegúrese de responder cada ítem
- Si no sabe cómo responder, simplemente tilde el casillero 3.

	Fuertemente en desacuerdo 1	**2**	3	4	**Fuertemente de acuerdo 5**
Pienso que me gustaría usar este producto frecuentemente.					
Encontré el producto innecesariamente complejo.					
Pienso que el producto es fácil de usar.					
Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.					
Encontré que varias funciones en el producto estaban muy bien integradas.					
Pienso que había mucha inconsistencia en el producto.					
Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.					
Encontré el producto muy incómodo de usar.					
Me sentí muy seguro usando el producto.					
Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.					

Figura 6.2: Cuestionario SUS enviado a cada participante

6.5. Análisis de resultados

A continuación, se muestran las respuestas y resultados obtenidos por cada participante:

Pregunta	Participante					
	1	2	3	4	5	6
Pienso que me gustaría usar este producto frecuentemente.	3	4	5	2	5	3
Encontré el producto innecesariamente complejo.	1	1	1	1	1	2
Pienso que el producto es fácil de usar.	3	4	5	3	4	4
Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.	2	3	1	1	3	2
Encontré que varias funciones en el producto estaban muy bien integradas.	3	3	4	2	5	3
Pienso que había mucha inconsistencia en el producto.	1	1	1	3	1	1
Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.	4	5	5	5	5	3
Encontré el producto muy incómodo de usar.	2	1	2	3	1	1
Me sentí muy seguro usando el producto.	3	2	5	4	3	4
Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.	1	2	1	1	1	3
Total	72,5	75	95	70	87,5	70
Promedio	78,3					

Figura 6.3: Cuestionario SUS - Resultados

Podemos ver que todos los resultados superaron los 65 puntos. Además, si realizamos un cálculo promedio obtenemos como resultado 78,3. Por lo tanto, podemos concluir que la escala de usabilidad del sistema es “Aceptable”.

6.6. Conclusiones

Finalmente se concluye que la carga de símbolos es sencilla e intuitiva, pero el modelado de reglas requeriría cambios para facilitar su uso. En particular se podría mejorar la UI presentada al usuario para agregar nuevos pasos, seleccionar múltiples tags en las condiciones, y describir cada tag con un ejemplo, que entendemos fueron los principales motivos en cuanto a si creían que necesitarían ayuda de un técnico para poder utilizar la herramienta. Se recibieron varios comentarios indicando que tener la posibilidad de testear la regla durante su modelado sería de gran ayuda para corroborar que este bien definida de acuerdo al objetivo buscado. Además la carga de los impactos podría hacerse al momento de definir el símbolo y no luego, en la página de edición del mismo. Luego, el resto de las funciones del sistema no presentaron inconvenientes a la hora de ser utilizadas por los participantes.

Capítulo 7

Manual de uso

La herramienta LELGenerator permite administrar los símbolos del LEL generado, las reglas que procesan texto para extraerlos, y una lista de símbolos descartados. La herramienta trabaja con un único LEL. Además se provee un chat para corroborar el funcionamiento de la herramienta. A continuación se detallan las funciones del sistema.

7.1. Pantalla principal

Desde esta pantalla se puede acceder al LEL generado, a la administración de reglas o a la lista de símbolos descartados, como se muestra en la figura 7.1. Además, en la parte superior se cuenta con una toolbar y un breadcrumb. En las próximas secciones se describirán estas funciones.

7.2. Chat

Al ingresar al chat el sistema solicita nombre y avatar del usuario, como se ve en la figura 7.2.

Una vez ingresado su nombre y avatar, se ingresa a la pantalla principal del chat, como se ve en la figura 7.3. Aquí se listan los mensajes enviados al chat, y la lista de usuarios conectados. Además, desde la barra superior se pueden acceder a las demás funciones del sistema.

Existen acciones dentro de las reglas que procesan los mensajes que pueden disparar interacciones del chatbot, preguntando a los usuarios del chat por nociones o impactos de símbolos detectados. Estos mensajes se visualizan en la figura 7.4.

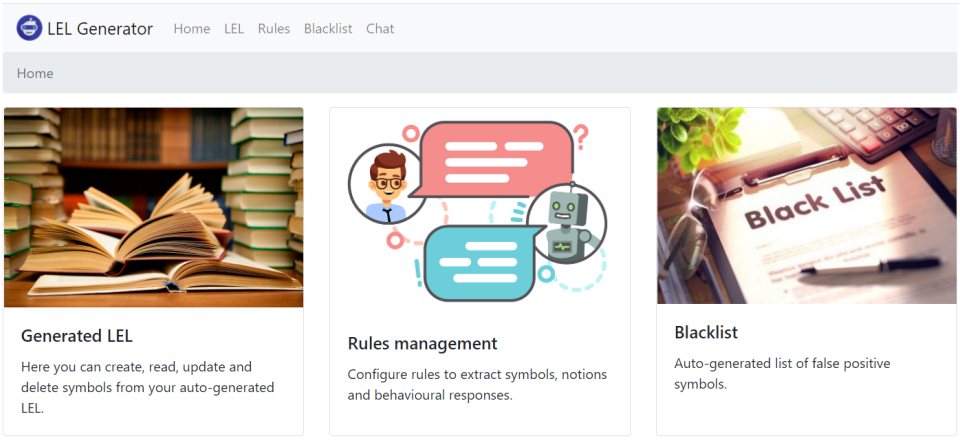

Figura 7.1: LELGenerator - Home

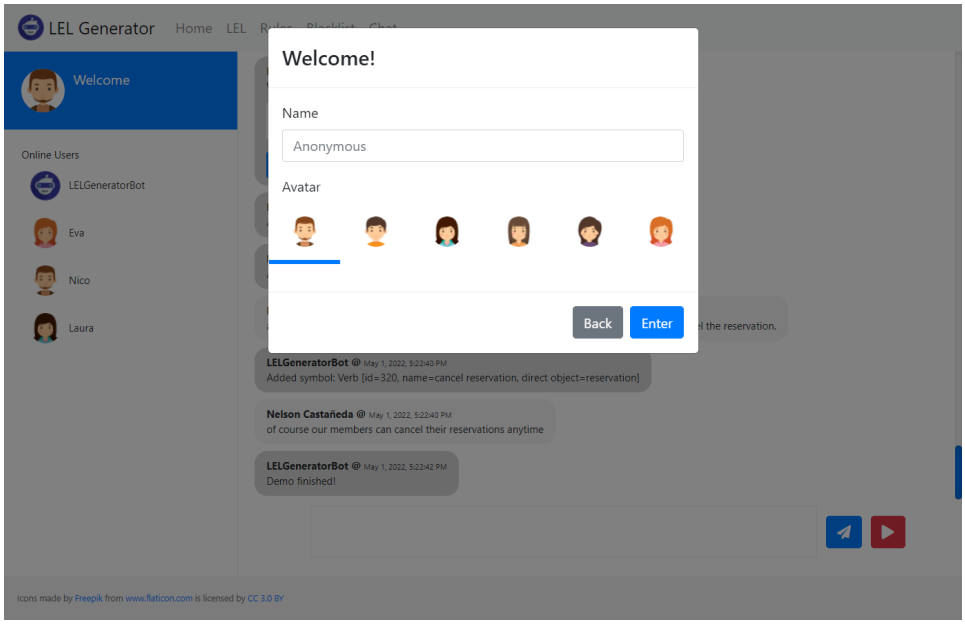

Figura 7.2: Chat de prueba - Datos iniciales

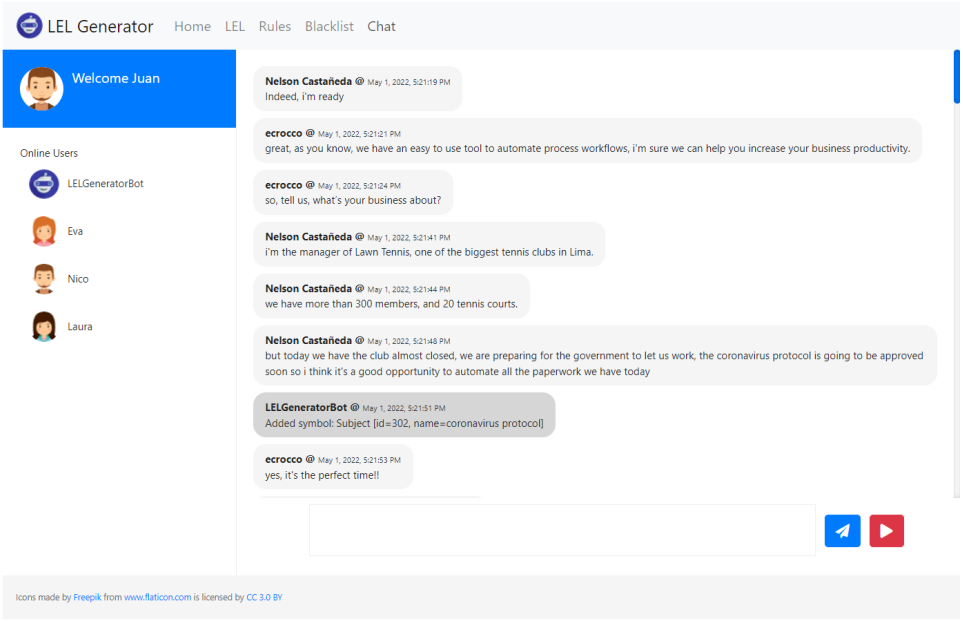

Figura 7.3: Chat de prueba - Pantalla principal

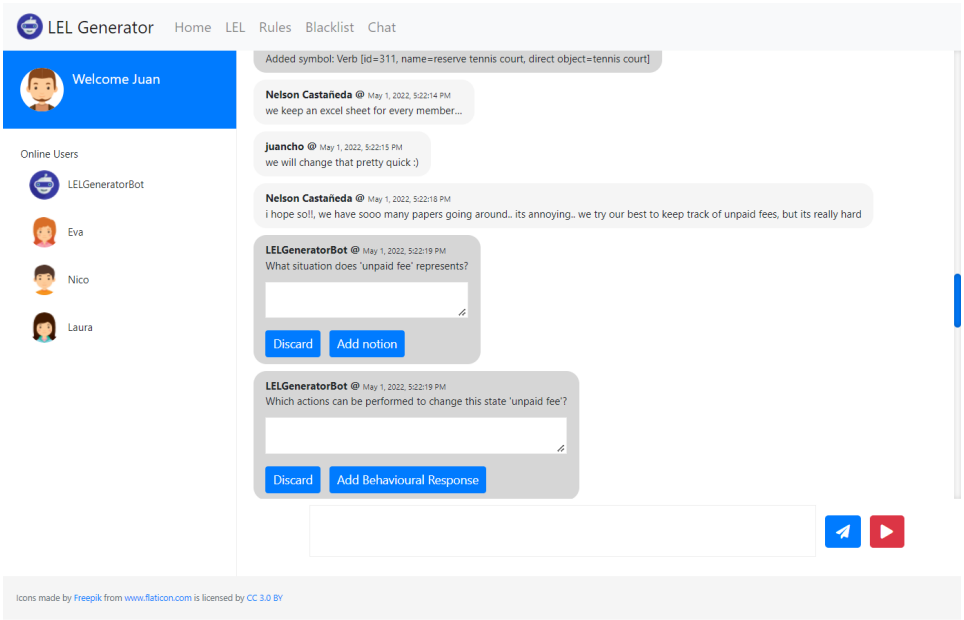

Figura 7.4: Chat de prueba - Preguntas del chatbot

7.3. LEL Generado

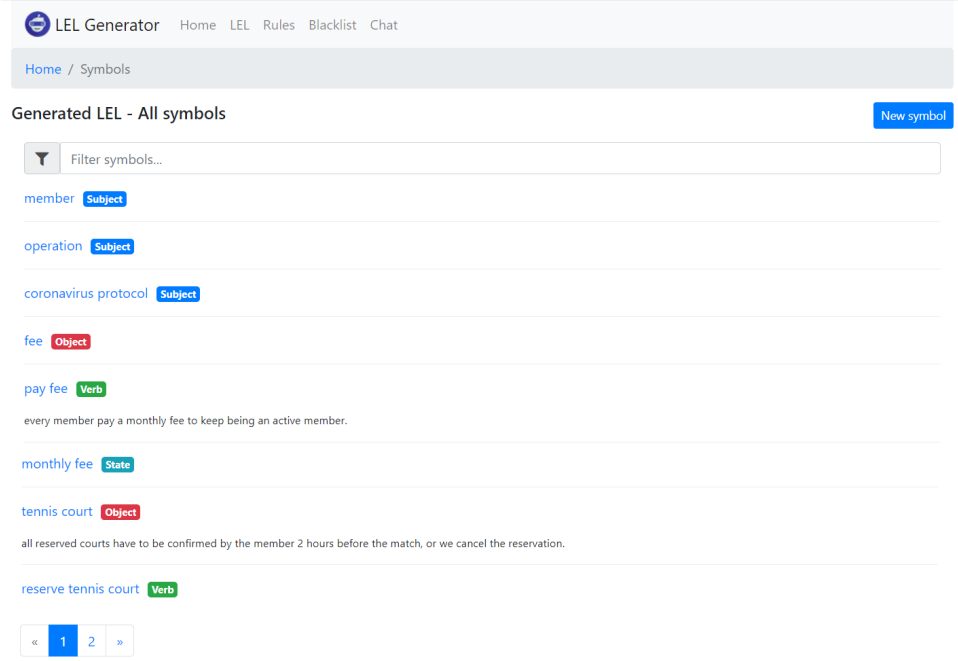

Figura 7.5: LELGenerator - Listado de simbolos del LEL

Al ingresar al LEL construido por la herramienta, se listan sus símbolos, como se muestra en la figura 7.5. Si se desea filtrar los símbolos por su tipo, se debe hacer click sobre la etiqueta que acompaña al símbolo, como muestra la figura 7.6.

Además, con el botón "New Symbol" el sistema desplegará una ventana para completar los datos del nuevo símbolo, como indica la figura 7.7. Por último, al pasar el cursor del mouse sobre la fila de un símbolo, aparecerá el botón "Delete" para eliminarlo. Para consultar un símbolo se debe hacer click sobre su nombre, de esta manera se accede a la consulta puntual de un símbolo.

7.3.1. Consulta de un Símbolo

En la pantalla que describe a un símbolo se pueden consultar su nombre junto a su tipo, su noción y sus impactos, como se ve en la figura 7.8. Cada

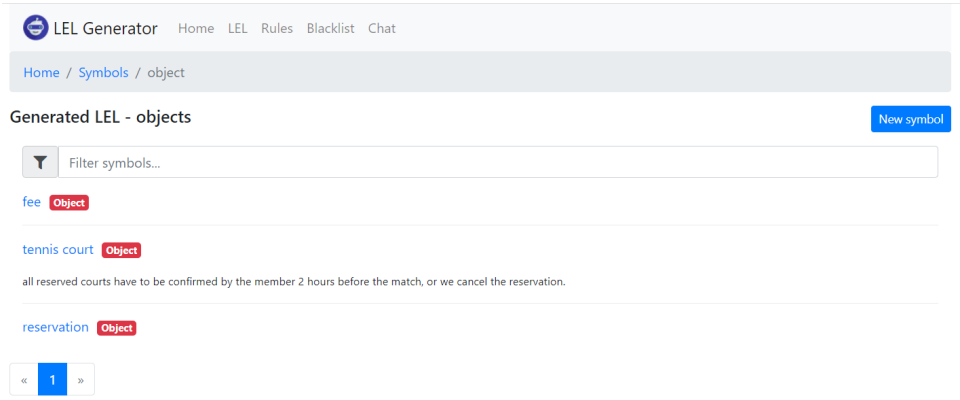

Figura 7.6: LELGenerator - Listado de Objetos del LEL

impacto que haga uso de otro símbolo permitirá consultarlo haciéndole click, lo que permite navegar al LEL con hipervínculos.

Se puede agregar la noción del símbolo usando el botón “Add notion”, en el caso que el símbolo ya tenga noción, esta puede ser eliminada usando el icono rojo a su lado. En el caso de querer agregar un impacto, se debe usar el botón “Add behavioral response”. Los impactos pueden ser eliminados usando el icono rojo a su lado. Además, por cada impacto, el sistema permite ir al momento en el chat en que se procesó dicha oración, como muestra la figura 7.9.

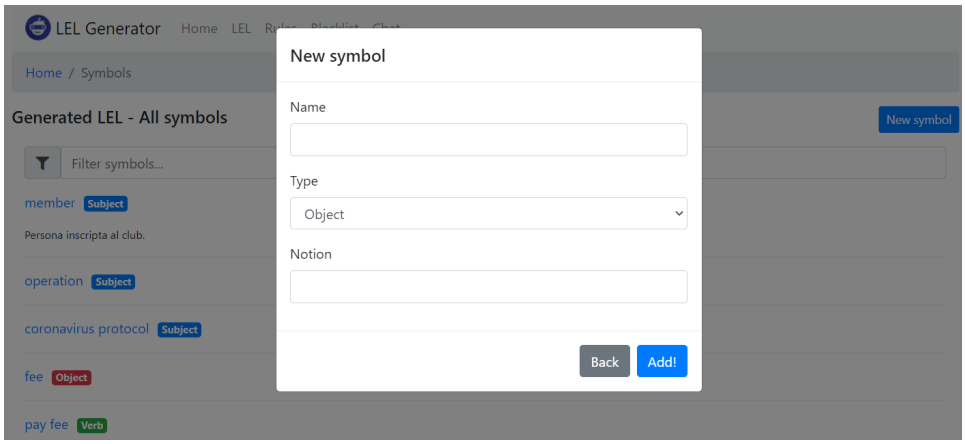

Figura 7.7: LELGenerator - Modal de nuevo símbolo

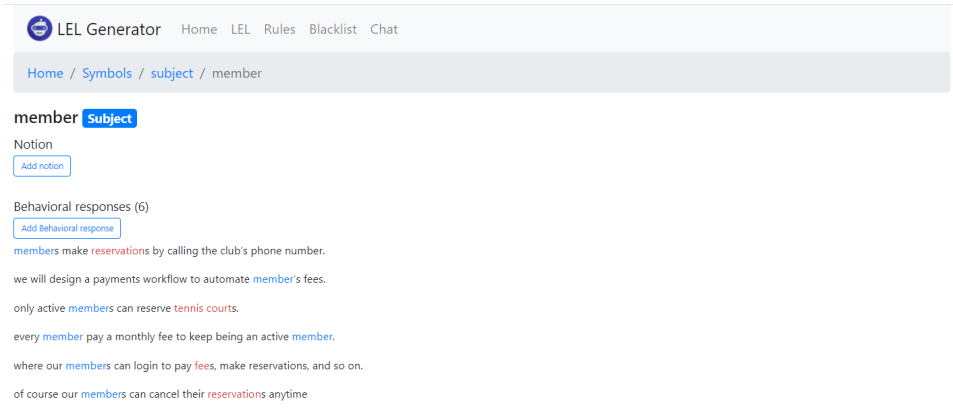

Figura 7.8: LELGenerator - Consulta del símbolo 'member'

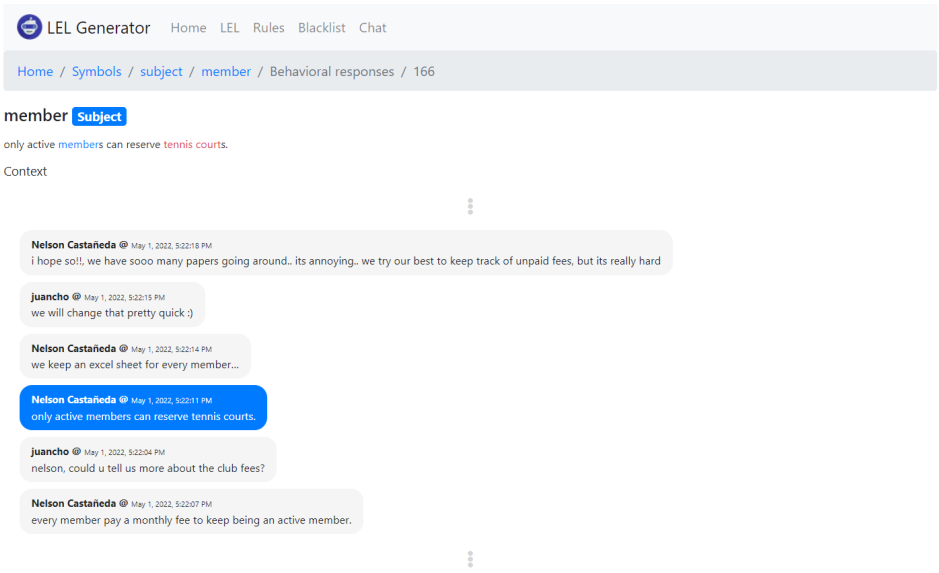

Figura 7.9: LELGenerator - Impacto del símbolo 'member'

7.4. Administración de reglas

Esta pantalla lista las reglas que utiliza el motor para procesar el texto. Por cada regla se indica su descripción y el listado de tags de las acciones que realiza, como se ve en la figura 7.10.

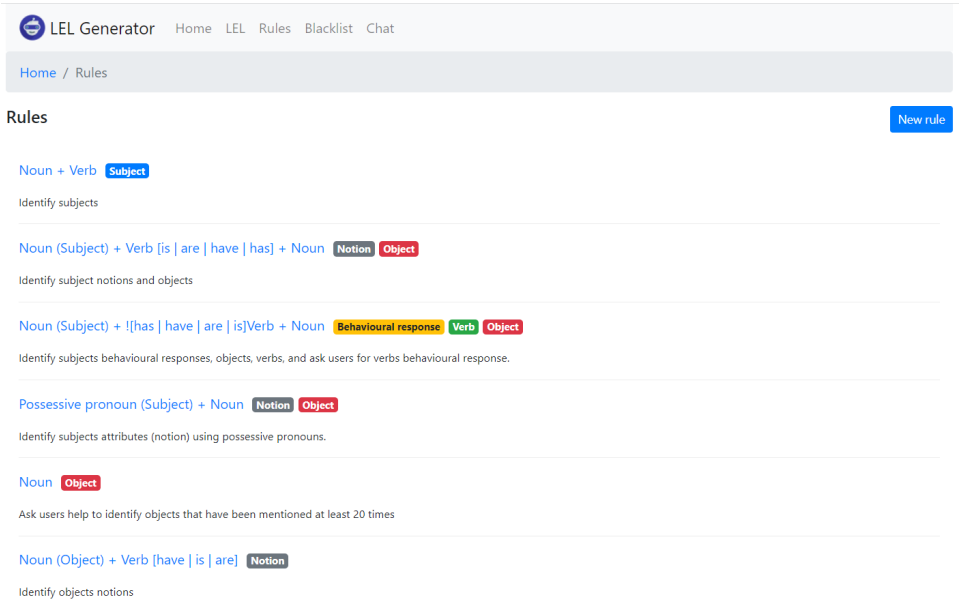

Figura 7.10: LELGenerator - Listado de reglas

Con el botón “New Rule” se accede a la pantalla de nueva regla, descrita en la sección 7.4.1. Además, al hacerle click sobre una regla se accede a la pantalla de consulta de regla. Por último, al pasar el cursor del mouse sobre una regla aparece el botón “Delete” para eliminarla.

7.4.1. Nueva regla

Para definir una nueva regla se debe especificar su descripción y los pasos de su ejecución. Para agregar nuevos pasos se debe utilizar el icono de +. El icono de cruz en la esquina superior de cada paso permite eliminarlo. Cada paso esta compuesto por condiciones y acciones. Para agregar alguno de estos, se debe utilizar el botón “Add”, que listará las opciones disponibles. Cada condición u acción puede ser eliminada usando la cruz que aparece a su lado. Una vez que la regla este modelada, con el botón “Create rule” se confirma la operación de alta.

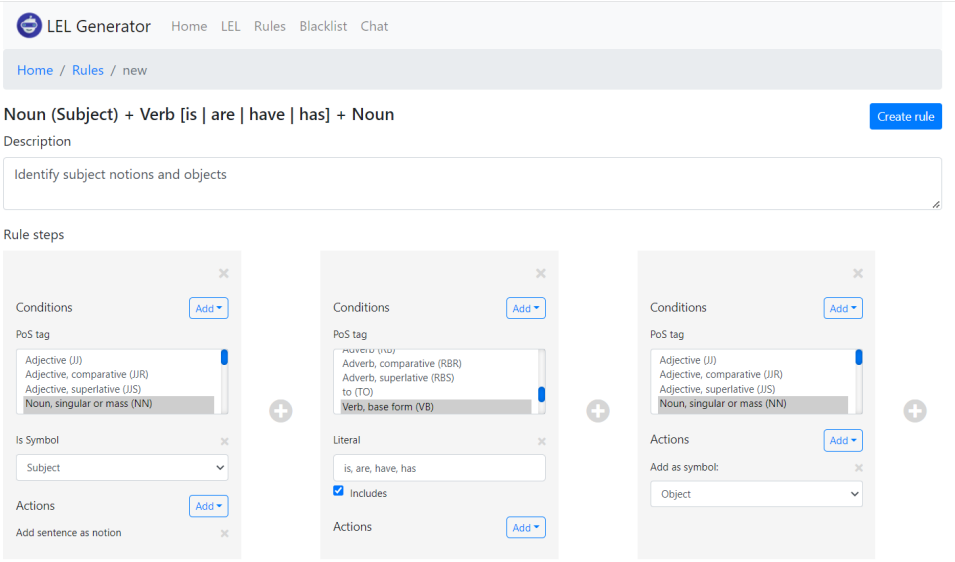

Figura 7.11: LELGenerator - Nueva regla

7.5. Blacklist

El usuario puede administrar los símbolos que fueron falsos positivos desde este apartado. Al ingresar, se muestra el listado como indica la imagen 7.5. El usuario puede ingresar nuevos símbolos a obviar usando el botón “New word”, o puede eliminarlos usando el botón “Delete”.

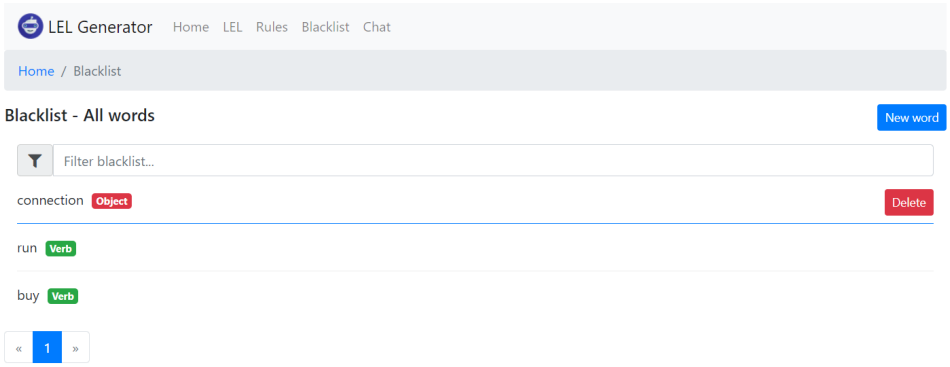

Figura 7.12: LELGenerator - Blacklist

Capítulo 8

Conclusiones y trabajos futuros

Terminando con la escritura de esta tesina, se realizará un breve repaso de los puntos destacados de este trabajo de fin de grado. Se comenzará con un resumen de los capítulos vistos hasta ahora y por último se finalizará con la conclusión que ha dejado este trabajo junto a posibles extensiones que mejoren la herramienta desarrollada.

Es importante que al iniciar el desarrollo de software los requerimientos necesarios sean lo más completos y correctos posibles. Comprender el contexto de un sistema de software durante la especificación de requerimientos es una tarea difícil y crítica. Todo aquello que no sea detectado durante esta etapa, o resulte mal entendido, provocará un gran impacto negativo en los requerimientos, propagando estos errores a las etapas sucesivas del desarrollo de software. Estas situaciones son más costosas de corregir cuanto más tiempo se haya avanzado en el proceso de desarrollo.

Definir el lenguaje de dominio antes de especificar los requerimientos es una manera de hacer frente a este problema. El LEL es un documento que permite definir el lenguaje de un dominio particular. Como se ha investigado, el uso de este documento es de gran apoyo para todas las etapas de la construcción de software, sin embargo, el esfuerzo que se debe hacer para construirlo impide que sea una técnica de uso habitual por los analistas. Esto nos motivó a analizar como podríamos facilitar su construcción, por lo que se inició la investigación del estado en el que se encuentra este campo.

De la investigación realizada, se encontraron herramientas que permiten construir el LEL con un costo relativamente bajo. La más interesante de ellas, plantea construirlo a partir del uso de técnicas de procesamiento de

lenguaje natural junto a un módulo de aprendizaje automático. Otra estrategia encontrada lo deriva a partir de las historias de usuario, a través del uso de un conjunto de reglas heurísticas. Estos enfoques requieren como entrada documentos completos, algo que no siempre se tiene al iniciar la etapa de especificación del software. Lo que si sabemos, es que se requiere de comunicación entre los analistas y los diversos stakeholders que sean involucrados en este proceso. Además, ambos trabajos no permiten la customización del análisis que realiza la herramienta sobre el texto, sino que los pasos para obtener los símbolos ya están codificados.

La herramienta construida permite a los analistas modelar la forma en que quieren que la herramienta realice el análisis del texto, permitiéndoles utilizar de manera implícita metadata obtenida de aplicar técnicas de procesamiento de lenguaje natural. Además fue concebida para ser utilizada dentro del contexto de un chat, entre los analistas y los stakeholders, agregando así un Bot a estas conversaciones, que es la voz de la herramienta. Esto permite que los analistas puedan indicar que bajo determinadas situaciones, el Bot participe del chat, preguntando que acciones tomar. La generación del documento LEL es incremental, se construye a medida que los mensajes son enviados. Además se provee una interfaz web para administrar el LEL capturado y modelar las reglas que el analista requiera.

Dentro de las principales fortalezas que tiene esta herramienta, es que procesa definiciones incompletas y sin ningún tipo de estructura, además de permitir modelar reglas que definen como es que los textos serán procesados. Este modelado tiene un grado de libertad inédito, que hasta el día de hoy no se encuentra en otras herramientas. Por último, se destaca el hecho de que fue diseñada y construida para poder ser integrada en aplicaciones de chat de uso masivo a futuro.

Para concluir este trabajo final, esta herramienta deja una base para que cualquier alumno pueda utilizarla y extenderla en futuros trabajos de grado. Se proponen algunas mejoras que podrían incorporarse, tales como:

- **Dependencias gramaticales:** Se podrían agregar nuevas condiciones que evalúen las relaciones gramaticales entre las palabras. Esto agregaría un poder descriptivo y de análisis superior al modelado de reglas.

- **Construir adaptadores a otras fuentes de chats:** En la actualidad la herramienta se conecta a un chat de ejemplo por medio de un adaptador. El diseño contempla la capacidad de agregar nuevos adaptadores, para conectar la herramienta a otras fuentes de chats.

- **Noción de sí misma:** Hoy la herramienta no entiende que es un símbolo, solo procesa las reglas modeladas por los analistas. Con un conjunto de reglas heurísticas simples, se podrían evitar cometer errores como por ejemplo, capturar un símbolo y categorizarlo erróneamente como Sujeto, cuando en realidad, no es una entidad que realice acciones (por que se trata de un objeto). Esto requeriría utilizar dependencias gramaticales.

- **Testear reglas durante el modelado:** Una falencia actual del modelado de reglas es que es difícil determinar si su funcionamiento es el correcto. Esto requiere tomar porciones de texto analizado y determinar si la regla debía evaluar verdadero o no. Es por esto, que sería de gran utilidad agregar al modelado la capacidad de testear la regla sobre un texto de ejemplo. De esta forma se podrían ver los metadatos obtenidos del análisis del texto y el resultado de haber sido evaluados por la regla.

Apéndice A

Anexo: Categorías gramaticales procesadas

Tag	Description	Used	Ignore	Reason
CC	Coordinating conjunction		X	Coordinating conjunctions, also called coordinators, are conjunctions that join, or coordinate, two or more items (such as words, main clauses, or sentences) of equal syntactic importance. In English, the mnemonic acronym FANBOYS can be used to remember the coordinators for, and, nor, but, or, yet, and so.
CD	Cardinal number		X	A cardinal number is a number used in counting to indicate quantity
DT	Determiner	X		For example: "He", "She".
EX	Existential there		X	The use of the expletive there in front of a verb—usually a form of be—to assert that someone or something exists. The construction as a whole is called an existential sentence. Existential there, also known as nonreferential there, is entirely different from there used as a place adverb: "It has no locative meaning, as can be seen by the contrast: There's a sheep over there. Also, existential there carries no emphasis at all, whereas the adverb does: There he is"
FW	Foreign word		X	
IN	Preposition or subordinating conjunction		X	A subordinating clause is a part of a sentence that adds additional information to the main clause. A subordinating conjunction is simply the word/words that is used to join a subordinating clause to another clause or sentence. I.e: "He was annoyed **because** the train had stopped."
JJ	Adjective	X		Adjectives are words that modify (describe) nouns. Adjectives do not modify verbs or adverbs or other adjectives.
JJR	Adjective, comparative	X		Comparative adjectives, make a comparison between two or more things.
JJS	Adjective, superlative	X		Superlative adjectives indicate that something has the highest degree of the quality in question. E.g: "The fastest car".

Tag	Description	Used	Ignore	Reason
LS	List item marker		X	
MD	Modal		X	
NN	Noun, singular or mass	X		Represents Objects and Subjects in the LEL.
NNS	Noun, plural	X		
NNP	Proper noun, singular	X		
NNPS	Proper noun, plural	X		
PDT	Predeterminer		X	Pre-determiners are normally placed before an indefinite article + adjective + noun to express an opinion about the noun they modify. Such and what are used to express surprise or other emotions.E.g: "such a good time" (Such is the PDT)
POS	Possessive ending	X		
PRP	Personal pronoun	X		We have both subject pronouns and object pronouns: Subjects: I, you, he, she, it, we, they. Objects: me, you, him, her, it, us, them.
PRP$	Possessive pronoun	X		Possessive pronouns show that something belongs to someone.
RB	Adverb	X		Adverbs tell us in what way someone does something. Adverbs can modify verbs (here: drive), adjectives or other adverbs. E.g: Mandy drives carefully.
RBR	Adverb, comparative	X		We can use comparative adverbs to show change or make comparisons: I forget things **more often** nowadays. We often use than with comparative adverbs: Girls usually work harder **than** boys.
RBS	Adverb, superlative	X		We can use superlative adverbs to make comparisons: His ankles hurt badly, but his knees hurt **worst**. When we intensify a superlative adverb, we often put the in front of the adverb: In our office, Jill works **by far the hardest.**
RP	Particle		X	In English grammar, a particle is a word that does not change its form through inflection and does not easily fit into the established system of parts of speech. Many particles are closely linked to verbs to form multi-word verbs, such as "go away."
SYM	Symbol		X	Symbols are "/" or "."
TO	*to*	X		Just the "To" word.
UH	Interjection		X	An interjection is a word added to a sentence to convey an emotion or a sentiment such as surprise, disgust, joy, excitement, or enthusiasm.
VB	Verb, base form	X		
VBD	Verb, past tense	X		
VBG	Verb, gerund or present participle	X		A gerund is formed by adding -ing to a verb. It functions as a noun
VBN	Verb, past participle		X	In English grammar, the past participle refers to an action that was started and completed entirely in the past. It is the third principal part of a verb, created by adding -ed, -d, or -t to the base form of a regular verb.
VBP	Verb, non-3rd person singular present	X		
VBZ	Verb, 3rd person singular present	X		

Apéndice B

Anexo: Símbolos y reglas a cargar en prueba de usabilidad

B.1. Símbolos

A continuación se describen los símbolos que serán cargados por los participantes de la prueba de usabilidad de la herramienta.

Name	Type	Notion	Behavioral responses
Process	Object	List of tasks a user has to complete in order to achieve a goal	-
Business Rule	Object	Where users can code their business logic	An administrator can execute business rules. An administrator can validate if a business rule work.
Form	Object	Digital representation of a paper form, composed of fields.	Users can complete a form by just clicking a button in the homepage.
Published	State	Indicates a Form is ready to be used	An administrator can publish a Form to have a published form.
Draft	State	Indicates a Form is under development	An administrator can create a Form to have a drafted form.
Execute rule	Verb	Action a user can make in the system to run the defined business logic	-
Validate rule	Verb	Users can validate their own business rules to check whether they are working using mock data.	-
Mock data	Object	Set of information for testing purposes	-
Administrator	Subject	Person responsible for carrying out the administration of a business or organization	-

Cuadro B.1: Listado de símbolos propuestos en la prueba de usabilidad

B.2. Reglas

A continuación se describen las reglas a modelar por los participantes de la prueba de usasbilidad de la herramienta.

B.2.1. Noun + Verb

Descripción: Identifica los símbolos de tipo sujeto.

Pasos:

- Paso 1

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)

  Acción: Agregar palabra como Sujeto.
- Paso 2

  Condición: PoS tag: Verb (VB, VBP, VBZ)

  Acción: Ninguna.

B.2.2. Possessive pronoun (Subject) + Noun

Descripción: Regla que captura la noción de un sujeto además de objetos del LEL.

Pasos:

- Paso 1

  Condición: PoS tag: Possesive pronoun (PRP)

  Condición: Es símbolo: Sujeto

  Acción: Agregar oración como noción del Sujeto.
- Paso 2

  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)

  Acción: Agregar palabra como Objeto.

B.2.3. Noun

Descripción: Regla que pregunta a los usuarios si una palabra es un Objeto del LEL

Pasos:

- Paso 1

Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
Condición: Cantidad de ocurrencias mayor a: 20
Acción: Preguntar al usuario si se trata de un Objeto del LEL.

B.2.4. Verb (Verb) + Noun + To

Descripción: Regla que identifica las nociones de un Verbo.

Pasos:

- Paso 1
  Condición: PoS tag: Verb (VB, VBP, VBZ)
  Condición: Es símbolo: Verbo
  Acción: Agregar oración como noción del Objeto.
- Paso 2
  Condición: PoS tag: Noun (NN, NNS, NNP, NNPS)
- Paso 3
  Condición: PoS tag: TO

Bibliografía

[1] Gelbukh. A. «Natural Language Processing and its Applications.» En: *Research in Computing Science* (2010), págs. 3-15.

[2] *Angular*. URL: `https://angular.io/` (visitado 22-05-2022).

[3] Rossi G. Antonelli L. Oliveros A. «Baseline Mentor, An Application that Derives CRC Cards from Lexicon and Scenario». En: *XXVIII JAIIO, II Workshop Iberoamericano en Ingeniería de Requerimientos, WER'99, Buenos Aires, Argentina* (1999).

[4] B.W Boehm. «Software Engineering». En: *Computer society Press, IEEE* (1997).

[5] Leite J.C.S.P. Breitman K.K. «Ontology as a Requirements Engineering Product». En: *Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE),IEEE Computer Society, Monterey Bay, California, USA, ISBN 0-7695-1980-6* (2003).

[6] John Brooke. «SUS: A quick and dirty usability scale». En: *Usability Eval. Ind.* 189 (nov. de 1995).

[7] Frederick Brooks Jr. *The Mythical Man-Month: Essays on Software Engineering*. Ene. de 1995. ISBN: 978-0-201-83595-3.

[8] Simon Brown. *C4-Model*. URL: `https://c4model.com/` (visitado 06-05-2021).

[9] Guillermo Federico Carrilao Ávila. «Construcción semiautomática de un documento LEL utilizando técnicas de procesamiento de lenguaje natural». Universidad Nacional de La Plata, 2021.

[10] Kevin Clark y Christopher D. Manning. «Entity-Centric Coreference Resolution with Model Stacking». En: (2015).

[11] Mike Cohn. *User Stories Applied: For Agile Software Development*. Ene. de 2004.

[12] Universal Dependencies contributors. *Universal Dependencies*. URL: `http://universaldependencies.org/docsv1/introduction.html` (visitado 11-10-2022).

[13] Oracle Corporation. *MySQL*. URL: `http://www.oracle.com/us/products/mysql/overview/index.html` (visitado 07-05-2021).

[14] K. Cox, M. Niazi y J. Verner. *Empirical study of Sommerville and Sawyer's requirements engineering practices, IET Software Volume: 3 , Issue: 5*. 2009, págs. 339-355.

[15] Leite JCSP Cysneiros L. «Using the Language Extended Lexicon to Support Non- Functional Requirements Elicitation». En: *proceedings of the Workshops de Engenharia de Requisitos, Wer'01, Buenos Aires, Argentina* (2001).

[16] Manning Christopher D. et al. «The Stanford CoreNLP Natural Language Processing Toolkit». En: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (2014), págs. 55-60.

[17] Alan Davis. *Just Enough Requirements Management: Where Software Development Meets Marketing*. Feb. de 2013.

[18] Marie-Catherine De Marneffe et al. «Universal dependencies». En: *Computational linguistics* 47.2 (2021), págs. 255-308.

[19] Oliveros A Gil D Figueroa D. «Producción del LEL en un Dominio Técnico.Informe de un caso». En: *In proceedings of the Workshops de Engenharia de Requisitos, Wer'00, Rio de Janeiro, Brazil* (2000).

[20] J. A. Moreiro González. «Perspectiva Documental del Procesamiento de Lenguaje Natural. Memorias Congreso SEPLN VIII, Universidad Carlos III, Madrid.» En: (1992).

[21] Festim Halili y Erenis Ramadani. «Web services: a comparison of soap and rest services». En: *Modern Applied Science* 12.3 (2018), pág. 175.

[22] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[23] Wiegers K. y Beatty J. *Software Requirements*. Ago. de 2013.

[24] Antonelli L. et al. «Buenas prácticas en la especificación del dominio de una aplicación». En: *Conferencia: XVI Workshop de Ingeniería en Requisitos, Montevideo, Uruguay* (2013).

[25] Antonelli L. et al. «Deriving requirements specifications from the application domain language captured by Language Extended Lexicon». En: *In proceedings of the Workshop in Requirements Engineering (WER), Buenos Aires, Argentina* (abr. de 2012), págs. 24-27.

[26] Antonelli L. et al. «Language extended lexicon points: Estimating the size of an application using its language». En: (2014).

[27] Leonardi C. y Leite J.C. y Rossi G. «Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural». En: *Tesis de maestría, http://www-di.inf.puc-rio.br/ julio/teses.htm, Facultad de informática, Universidad Nacional de La Plata, Argentina* (2001).

[28] Franco A.P. Leite J.C.S.P. «Uso de Hipertexto na Elicitaçao de Linguagens da Aplicaçao». En: *IV Simpósio Brasileiro de engenharia de software Sau Pedro* (1990).

[29] Gavaldá. M. «La investigación en tecnologías de la lengua. Research in language technology». En: (2011).

[30] Sam Mclellan, Andrew Muddimer y S. Peres. «The Effect of Experience on System Usability Scale Ratings». En: *Journal of Usability Studies* 7 (nov. de 2011).

[31] Y. Mizuno. «Software Quality Improvement». En: *IEEE Computer, Vol. 16, No. 3* (mar. de 1983), págs. 66-72.

[32] Paul Murley et al. «Websocket adoption and the landscape of the real-time web». En: *Proceedings of the Web Conference 2021.* 2021, págs. 1192-1203.

[33] *Penn Treebank tagset.* URL: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html` (visitado 13-05-2021).

[34] Matıas Sebastián Sarmiento y Mariela Fernanda Zurbano. «Aplicación web para la construcción colaborativa del Léxico Extendido del Lenguaje». Universidad Nacional de La Plata, 2019.

[35] Kim Schmiedehausen. «Single page application architecture with angular». En: (2018).

[36] *SockJS.* URL: `https://github.com/sockjs/sockjs-client` (visitado 22-05-2022).

[37] Thomas Stahl et al. *Model-Driven Software Development: Technology, Engineering, Management.* Ene. de 2006. ISBN: 978-0-470-02570-3.

[38] *Stomp*. URL: https://github.com/stomp-js/stompjs (visitado 22-05-2022).

[39] Stanford University. *Coreference Resolution*. URL: https://stanfordnlp.github.io/CoreNLP/coref.html (visitado 11-10-2022).

[40] Stanford University. *Stanford Dependency Parsing*. URL: https://nlp.stanford.edu/software/stanford-dependencies.html (visitado 11-10-2022).

[41] Matias Urbieta et al. «The impact of using a domain language for an agile requirements management». En: *Information and Software Technology* 127 (2020), pág. 106375.