UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO-SALAMANCA
División de Ingenierías

"Análisis de oraciones en el idioma inglés usando procesamiento del lenguaje natural."

TESIS:

QUE PARA OBTENER
EL GRADO DE MAESTRA EN INGENIERÍA ELÉCTRICA

PRESENTA:
Isabella Ojeda Núñez

DIRECTOR:

Dr. Sergio Eduardo Ledesma Orozco
M. Sc. Ana Isabel Gallardo García

Salamanca, Guanajuato. Febrero, 2018

Agradecimientos

A mi madre y a mi padre por formarme durante todos estos años para que cada día sea una mejor persona y lograr metas como ésta. En esta etapa en particular le agradezco a mi madre por animarme y darme fortaleza, a mi padre por escucharme, darme consejo y ayudarme a analizar las situaciones desde diferentes perspectivas, y a ambos por todo el amor y paciencia aun en los momentos mas difíciles.

A mis hermanos y tíos Maria E, y Alfredo por seguir presentes en mi vida con su apoyo, cariño y su compañía aun en la distancia.

A Gustavo por ser desde el primer momento un gran amigo y apoyo durante esta etapa de mi vida, por sus consejos, por escucharme y por hacerme ver las cosas que muchas veces no podía ver o aceptar por mi misma. Por su cariño y el de su familia que siempre me hizo sentir mas en casa.

A mis amigos Venezolanos y Leidy por todos los momentos vividos, algunos buenos y otros malos pero todos necesarios para crecer y valorar mas su amistad y cariño, y aprender que puedes tener una especie de familia sustituta.

A los profesores, Ledesma por todo su apoyo durante mi trabajo de tesis y por darme la oportunidad de acudir a su clase y hacer mi estancia, la maestra Gallardo quien también me guio en mi tesis, Camarena y Villafaña por compartir sus conocimientos.

Finalmente, a Conacyt y a la Universidad de Guanajuato por darme la oportunidad de hacer esta maestría y obtener mas conocimientos para mi vida profesional y personal.

Índice general

Capítulo 1

Introducción

El propósito de esta tesis consiste en realizar el análisis de oraciones en el idioma inglés, para obtener información a partir de dichas oraciones. Los datos que se extraen a partir de las oraciones son las respuestas a tres de las preguntas conocidas en inglés por la abreviatura "5 Ws", que se refiere a what, who, when, where, why, (en español: qué, quién, cuándo, dónde y por qué). En este trabajo se da solución a las tres primeras preguntas, siendo estas, de acuerdo con el autor, las que brindan la información más relevante.

Además de esto, también se realiza la ubicación de un tipo de error gramatical dentro de las oraciones en caso de que exista. Esto se logra poniendo en práctica alguna de las reglas de la gramática inglesa.

Para realizar el proceso de análisis se utiliza programación orientada a objetos. El proceso de análisis consiste principalmente en tres etapas, estas etapas son:

- Análisis Lexicográfico
- Clasificación de la oración en tipos de frases
- Ubicación de la información requerida

1.1. Panorama General

Actualmente, gracias a la tecnología y en su mayoría a la Internet, el ser humano tiene acceso a grandes cantidades de información de todo tipo. Está claro que mientras más información posea el ser humano mayor conocimiento se puede tener. Pero muchas veces cuando se tiene a la mano gran cantidad de datos, no se sabe por dónde empezar a procesarla o cómo manejarla, lo que ocasiona que sea agobiante para el usuario.

La mayoría de la información se encuentra en un formato digital y es procesada por los dispositivos electrónicos para que el ser humano pueda acceder a ella y entenderla. Debido a la forma en que se encuentra la información y a la gran cantidad que existe, se buscan y se desarrollan herramientas para que el manejo de la información sea mucho más fácil para el ser humano.

En esta ardua búsqueda, se ha venido estudiando el procesamiento del lenguaje natural, que consiste en darle la capacidad a las máquinas para entender la información, en formato de lenguaje natural, como lo hacen los humanos y no sólo cómo una cadena de unos y ceros [1].

El procesamiento del lenguaje natural es una de las ramas que se ha estudiado a lo largo de los años para ayudar al ser humano a manipular información expresada en lenguaje natural, ya sea para acelerar las búsquedas, manipular texto de forma automática, hacer comparaciones de textos, extraer información clave [2] entre muchas otras aplicaciones. El objetivo que pretende lograr el procesamiento del lenguaje natural es procesar la información de una forma más coherente y más cercana a como lo haría un humano, pero en menor tiempo, pues la máquina puede procesar información mucho más rápido que el ser humano.

1.2. Objetivos

1.2.1. Objetivo General

Analizar oraciones en inglés utilizando procesamiento del lenguaje natural a través de la programación orientada a objetos.

1.2.2. Objetivos específicos

- Realizar un analizador lexicográfico para extraer los componentes básicos de las oraciones (tokens).
- Realizar la búsqueda de al menos un tipo de error gramatical dentro de las oraciones analizadas e indicar la aparición de éste en caso de presentarse.
- Desarrollar un algoritmo que permita la extracción de información relevante de las oraciones para responder a las preguntas más relevantes, tales como quién, qué y cuándo.
- Validar la capacidad de análisis del algoritmo a través de oraciones de textos de diferentes niveles de dificultad en el idioma inglés.

Para el desarrollo de estos objetivos se hace necesario el conocimiento, investigación y estudio de:

- Programación orientada a objetos en C++.
- Reglas gramaticales del inglés.

1.3. Justificación

Con el transcurso de los años, los seres humanos se han rodeado de máquinas cada vez más óptimas para ayudarse en sus tareas. Estas máquinas no manejan el lenguaje de los

humanos, sino un lenguaje propio, binario. Así pues, la comunicación con la máquina no es igual que cuando se comunican los seres humanos entre sí.

El lenguaje natural, independiente del idioma, es lo que utilizan los humanos comúnmente para comunicarse, es el lenguaje con el que se observa que se logra mayor y más fácil entendimiento [3]. En este sentido, la gran mayoría de la información y conocimiento que se tiene al alcance se guarda y se maneja en forma de lenguaje natural [1].

En este orden de ideas, se ha buscado integrar de la mejor manera el lenguaje natural a las máquinas, por ello surgió la disciplina del procesamiento del lenguaje natural. Que busca que las computadoras sean capaces de entender este lenguaje y procesarlo de alguna manera.

Al permitir que las máquinas puedan manejar el lenguaje natural se abre una infinidad de aplicaciones, como las industriales, las cuales pueden optimizar una tarea y reducir el tiempo de trabajo en dicha tarea hasta un 40 % [4], reducir el tiempo de búsqueda de un investigador al utilizar herramientas que le permitan extraer frases claves de los textos [2].

Actualmente también existen buscadores para encontrar información específica en la nube de manera rápida, pero muchas veces estos buscadores se basan en conseguir el mayor número de coincidencias de palabras en los textos y en función de esto mostrar una respuesta a la búsqueda, con el procesamiento del lenguaje natural, se pueden optimizar estas búsquedas y aproximarse cada vez más a la información que realmente desea obtener el usuario [1].

A través del procesamiento del lenguaje natural y las redes sociales también es posible saber que eventos están sucediendo en tiempo real, cual es la postura o sentimiento de la gente ante la situación [5] y con dicha información tomar alguna acción.

El procesamiento del lenguaje natural además es una herramienta que se utiliza en otras áreas como es el caso de la minería de texto, la cual es una rama parecida a la minería de datos, pero en este caso toma la información objeto de estudio de textos, como por ejemplo de redes sociales, tal como Twitter [6].

Cada vez existe más información al alcance de todos, y por ello es sumamente apremiante explorar las áreas que permitan el manejo de dicha información de manera óptima. Esta rama puede lograr, entre otras cosas, que se automaticen actividades, se gestione mejor la información, se realicen seguimientos de tendencias o gustos para posteriormente reducir horas de trabajo, recursos económicos, mejorar redacciones de textos y otras muchas implementaciones. Es por ello que merece ser estudiada y aplicada.

En este trabajo se aplica el procesamiento del lenguaje natural para extraer información relevante de una oración. Todo los textos están formados por infinidad de oraciones, por ello se puede decir que las oraciones son la base de todo texto. Al lograr analizar una oración y extraer la información requerida, se puede posteriormente extrapolar los algoritmos con más detalle y más estructuras para abarcar un texto completo.

1.4. Contribuciones

Esta tesis contribuye al desarrollo de la inteligencia artificial, debido a que el procesamiento del lenguaje natural es una rama de esta ciencia.

A través de este documento se puede entender un poco más la forma de abordar el procesamiento del lenguaje natural, pues debido a la complejidad que tiene no es una rama que sea sencilla de afrontar. No se pretende hacer un estándar de cómo abordarlo, pero el lector se puede dar una idea de lo que debe considerar antes de empezar a trabajar en el área.

1.5. Estructura del documento

Este documento está divido como se describe a continuación:

En el Capítulo 2 se define el procesamiento del lenguaje natural y se hace un recorrido por los diferentes trabajos que se han realizado en el área. Además se explican sus implementaciones, aportes y logros.

Posteriormente se definen y plantean los conceptos claves y relevantes para la investigación, así como las teorías existentes y las implementadas.

En el Capítulo 3 se describe la metodología implementada para realizar esta investigación. Se especifican y describen las diferentes etapas y procesos que se siguieron para lograr los objetivos.

En el Capítulo 4 se muestran y explican los resultados obtenidos de acuerdo a los objetivos planteados. Además, se ilustran algunos ejemplos de las pruebas realizadas.

El Capítulo 5, último de este documento, contiene las conclusiones derivadas de esta investigación, así como algunas recomendaciones para trabajos posteriores en esta área de investigación.

Capítulo 2

Fundamentos Teóricos (Trabajos Relacionados)

El procesamiento natural del lenguaje es un área que se ha investigado desde 1940 [7]. Debido a su complejidad se le han dado diferentes enfoques algunos dando mejores o peores resultados, muchas veces también se elige un enfoque según su aplicación, las cuales son bastante diversas.

En este capítulo, en la primera sección se mencionan alguno de estos enfoques y de las aplicaciones que se le han dado. Posteriormente se definen los conceptos y herramientas utilizadas para el desarrollo de esta investigación.

2.1. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural es una ciencia o disciplina que se encarga de estudiar y desarrollar mecanismos que permitan a las computadoras entender y procesar textos o discursos en lenguaje natural, de la forma en que lo hacen los humanos y no simplemente de forma binaria como lo hace comúnmente una máquina. Con ello se busca que las computadoras sean capaces de manipular el lenguaje natural para extraer información, ofrecer interfaces en lenguaje natural, realizar traducción automática, optimizar los buscadores, reconocer discursos entre muchas otras aplicaciones [8].

Es un área que abarca varias disciplinas entre ellas las matemáticas, sistemas computacionales, lingüística, probabilidad, inteligencia artificial, psicología [9]. Esta misma diversidad hace que sea una ciencia que se estudia y se desarrolla de diferentes formas con el objetivo de buscar los mejores resultados.

Actualmente existen diferentes enfoques para el procesamiento del lenguaje natural (PLN, por sus siglas en español), uno de ellos es el método estadístico [10]. Este enfoque parte de la premisa de que el conocimiento humano es probabilístico, y por ende el lenguaje también lo será. Por ello para poder manejar y entender la teoría del lenguaje de una forma más exacta, la probabilidad debe ser una parte importante del PLN. Algunos investigadores consideran

que con este enfoque, se pueden resolver con mayor precisión problemas de ambigüedades que se presentan en ocasiones en el lenguaje natural. Otro enfoque que se toma para el PLN es el lingüístico, este se basa solamente en las reglas y formalidades que describen el lenguaje natural [10].

Este último enfoque que se menciona, es conocido por muchos nombres, entre ellos gramática libre de contexto (contex-free grammar), base de muchos de los modelos de la sintaxis del lenguaje natural. Este enfoque maneja lo que se conoce como *constituyentes*, que no es más que un arreglo de palabras que se comportan como una unidad en la oración, como por ejemplo una frase sustantiva. Esta idea de basar la gramática en constituyentes viene desde 1900, cuando el psicólogo Wilhelm Wundt (1900) lo propuso, pero fue formalizada por Chomsky en 1956 [11].

Tal como se mencionan, existen varios métodos para lograr el procesamiento del lenguaje natural, pero a pesar de esto, casi todos los sistemas de PLN presentan ciertas similitudes en cuanto a sus componentes esenciales. Los componentes en común son la gramática o fuente de datos, lo que representa el conocimiento previo del sistema, un algoritmo que analiza las oraciones y le asigna una o más etiquetas de clasificación, un analizador sintáctico y finalmente algún tipo de acción [7], [3].

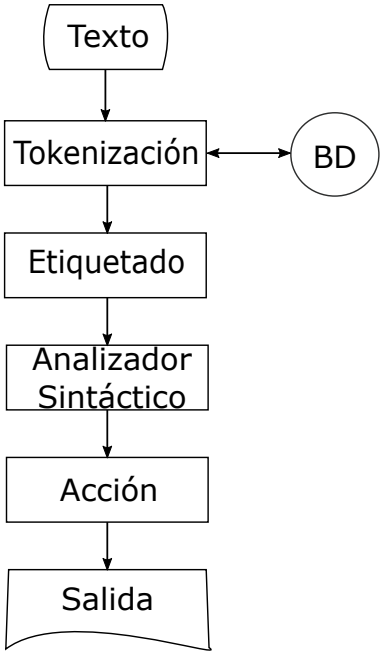


Figura 2.1: Componentes esenciales de un sistema de PLN

La Figura 2.1 muestra de forma resumida lo mencionado en el párrafo anterior. Un texto entra al sistema y dicho texto pasa por una serie de procesos. Primero se *tokeniza*, es decir se identifican los diferentes grupos de caracteres que forman una palabra, luego se aplica algún tipo de etiquetas a dichas palabras, posteriormente con los *tokens* y sus respectivas

etiquetas se hace un análisis sintáctico y con la respuesta que este análisis arroje se ejecuta alguna acción en particular que va a generar una salida final.

Una de las herramientas utilizadas para la fuente de datos de los algoritmos de procesamiento del lenguaje natural son los *corpus lingüísticos*. Los "corpus", son un compendio de fragmentos de textos, extraídos de fuentes consideradas fiables, textos escritos en un lenguaje natural adecuado, con buena redacción y coherencia. Dependiendo del *corpus* estos pueden ser de una o diversas áreas.

Los *corpus* se han creado desde hace muchos años atrás y actualmente se recopilan en forma digital. Por mencionar algunos está el *Linguistic Data Consortium* (LDC) y el *Brown corpus* [10]. La mayoría de ellos se pueden utilizar pagando una pequeña cantidad de dinero.

Debido a que los *corpus* son colecciones de fragmentos de textos, puede ser que el mismo no sea una muestra representativa del área de aplicación. Una muestra es representativa si lo que se define en la población de muestreo es aplicable para la población en general. No hay una forma fácil de definir si un *corpus* es o no representativo [10]. En vista de esto, se observa que utilizar un *corpus* tiene sus desventajas al no poder definir qué tan representativo es en el entorno en donde se quiere aplicar el PLN.

Otra cosa muy común que también sucede con los *corpus*, es que los fragmentos de textos extraídos, generalmente, son de textos antiguos y muy formales. Estos dos aspectos ocasionan que no se pueda utilizar dicho *corpus* para la base de datos de un algoritmo de PLN que se quiere aplicar a temas de actualidad o en entornos como las redes sociales donde las palabras y la forma en que se escribe no es la misma que se va a conseguir en un documento formal. Al usar estos *corpus* en dichos entornos puede ser que no se tengan los resultados esperados pues no será la base de datos más adecuada. Es por esta razón que se busca otras alternativas para crear las bases de datos a utilizar por el PLN, como por ejemplo la creación de bases de datos propias.

2.2. Estado del arte

Actualmente hay un gran número de trabajos en el área del procesamiento del lenguaje natural pues se ha visto el potencial que este puede ofrecer. Los trabajos realizados en PLN abarcan diversas áreas, como la medicina, la industria, en veterinaria sólo por nombrar algunas.

En medicina el PLN se ha utilizado para resolver áreas complejas como es el caso de la transducción de señales, envío de señales químicas o físicas a través de las células. En esta área en particular utilizaron la PLN para conseguir información relevante de la web y de recursos literarios y extraer dicha información para luego ser utilizada por un agente con el fin de generar hipótesis sobre la transducción de las señales [12].

"Stanford CoreNLP" es una herramienta basada en Java, que contienen la mayoría de los componentes de una herramienta de PLN y es de software libre. Su primera versión fue desarrollada en 2006. Sirve para el análisis del lenguaje natural; al revisar el texto agrega algún tipo de análisis de información al texto. Entre la información que puede brindar se encuentra la forma básica de una palabra, indicar sentimientos e indicar cuáles oraciones sustantivas se refieren a la misma entidad.

Esta herramienta puede ser invocada desde escenarios comunes para realizar análisis del lenguaje, y puede ser aplicada a un párrafo, así como también a un texto completo. En principio se realizó para el idioma inglés, pero se han desarrollado paquetes para que sea compatible con idiomas como árabe, chino, francés y español [13].

Como se mencionó anteriormente, una parte de la información que puede brindar *Standford CoreNPL* es indicar sentimientos, existen investigaciones que sólo se han enfocado a este tipo de análisis y por ello brindan más información del texto al no clasificar todo el texto como positivo o negativo, sino a clasificar diferentes tópicos del texto como positivo o negativo. Estos trabajos han demostrado entre 75 y 95 %, de precisión dependiendo de los datos, en páginas web y artículos de periódicos. Tal es el caso del trabajo presentado por Tetsuya Nasukawa y Jeonghee Yi [14].

Las investigaciones para indicar sentimientos en los textos son diversas, y algunos se han atrevido a ir más allá, como es el caso de [15] donde identifican frases sarcásticas a través de los contrastes que existen en dichas frases cuando se menciona un sentimiento positivo en una situación negativa. En esta investigación desarrollan un algoritmo que aprende de forma automática frases que se refieren a sentimientos positivos y frases de situaciones negativas, después utilizan las frases aprendidas para identificar sarcasmos en *tweets*.

En el 2014 en la conferencia de la asociación de computación lingüística se presentó "DKPro Keyphrases", la cual es una herramienta que permite la extracción de las frases claves de un texto, este programa las extrae, las prioriza y posteriormente las evalúa para arrojar los resultados. Es una herramienta que puede ayudar a los investigadores para conseguir información relevante de forma automática. Además, esta herramienta ofrece un marco de referencia de la programación para los investigadores interesado en desarrollar nuevas herramientas de este tipo [2].

En veterinaria, también se ha aplicado la PLN. En esta área se realizó una herramienta capaz de diferenciar en publicaciones de foros veterinarios, si lo publicado se refiere a un caso de un paciente en particular o si se refiere a preguntas en general. Para ello utilizaron una pequeña base de datos que ya contenía anotaciones sobre casos de pacientes y con esta entrenaron un sistema de extracción para obtener los atributos de un paciente veterinario. Luego con este sistema de extracción aplicado a un gran número de textos sin anotaciones crearon un léxico de atributos de pacientes veterinarios [16]. Posteriormente este léxico se utilizó en el clasificador que distingue entre una pregunta general de tipo veterinaria o un caso de un paciente.

En el ámbito industrial la PLN también se ha abierto campo, al menos esto se puede inferir de la investigación realizada por Per Runeson, Magnus Alexandersson y Oskar Nyholm, en la cual realizan la detección de reportes duplicados de fallas utilizando en procesamiento del lenguaje natural. Básicamente lo que hacen es tomar la información en texto plano, hacerle procesamiento al texto y luego utilizan estadísticas para verificar el número de veces en que aparecen las palabras para identificar reportes duplicados. En las pruebas que han realizado han podido identificar dos de tres reportes duplicados [4], esto es un buen resultado que se pude ir mejorando con la inclusión de otras técnicas o utilizando algún otro método.

Además de estas investigaciones que se basan en lenguaje natural escrito, también existen otras investigaciones en las cuales se toma como entrada lenguaje verbal. Este es el caso de Articulate, el cual es un sistema que permite a usuarios con poco conocimiento de las herramientas de los software de visualización, realizar una representación visual de los datos de manera eficaz [17], manejando el software con lenguaje natural en lugar de procedimientos complicados.

Los métodos utilizados para la PLN también son variados, esto debido a su complejidad, a la gran cantidad de excepciones y ambigüedades que se presentan en el lenguaje. Para solucionar los diversos problemas que se presentan los investigadores han tratado de abordar la PLN de diversas formas para conseguir la más óptima.

Varios autores utilizan máquinas de estado finito para trabajar la PLN [18], [19]. Según el autor estos algoritmos basados en máquinas de estado mejoran el espacio y el tiempo de procesamiento [18], además ayudan a codificar la gran cantidad de reglas que rigen la morfología léxica. Se puede decir que el léxico es infinito, muchas veces existen palabras que no se consiguen en el diccionario y pueden ser palabras base con un sufijo o prefijo, una máquina de estado puede resolver este problema y evitar errores [20].

Sin embargo, otros autores, piensan que una máquina de estado no es una solución adecuada para abordar la PLN si este se maneja a través de reglas gramaticales, pues generalmente estas reglas presentan recursión, algo que no puede manejar una máquina de estado finito [11]. En esta investigación se utiliza un método que si permite utilizar la recursión y por ende se pueden implementar las reglas gramaticales de la sintaxis en las oraciones, es decir las reglas que definen como se forma una oración.

Se puede observar la gran cantidad de trabajos y técnicas que existen para la PLN. Pero algo que se observa de forma recurrente es que la mayoría de los trabajos se enfocan en un área o tema en particular, esto puede deberse a diferentes aspectos, como por ejemplo, la complejidad que presenta el lenguaje natural al momento de intentar crear algoritmos para que la máquina pueda procesarlo, el vocabulario tan extenso que existe en cualquier idioma, la evolución de la lengua, las diferentes formas de escribir o hablar dependiendo del entorno, entre muchas otras.

2.3. Lenguaje

El lenguaje es un sistema de comunicación a través del cual los humanos expresan sus deseos, necesidades, opiniones y pensamientos. El lenguaje natural se compone de símbolos, que forman palabras, estas a su vez se combinan para formar oraciones, y todo en conjunto forma el discurso.

Los humanos son capaces de formar y entender un número infinito de oraciones, incluso sin haberlas escuchado con anterioridad. Sin embargo esto no significa que el humano sea capaz de almacenar un número infinito de oraciones en su cabeza, algo más debe suceder para que el ser humano tenga esta capacidad. Se piensa que esto es posible porque el hombre conoce y tiene en su cabeza una serie de reglas que rigen la forma en que crea o entiende las diferentes oraciones que existen en el lenguaje que utiliza [21].

Esta hipótesis ha sido aceptada por la mayoría de los lingüistas, quienes se encargan de estudiar el lenguaje a través de diferentes teorías. Esta teoría es conocida como gramática generativa, definida como aquélla que puede generar un número infinito de oraciones correctamente formadas a partir de un número finito de reglas o principios [22], esta serie de reglas se conoce como gramática.

En esta investigación se toma como punto de partida este enfoque para desarrollar los algoritmos necesarios para hacer PLN.

2.4. Gramática

Puesto que en esta investigación se plantean algoritmos para procesar lenguaje natural, se hace necesario manejar ciertos términos importantes de la lengua, como lo es la gramática, las palabras, clasificación de las palabras y otros. La gramática es la ciencia que estudia entre otras cosas, las clases de palabra, su función y relación en las oraciones y la sintaxis; la estructura de las oraciones.

Las palabras, unidad básica de la sintaxis, son un conjunto de caracteres alfanuméricos con espacios en ambos lado, estos caracteres pueden tener apóstrofes y guiones (como por ejemplo *sugar-free*) [23] pero ningún otro signo de puntuación [10]. Este es un concepto un poco limitado, pues vemos que existen palabras unidas por un guión y palabras seguidas de un punto o una como, es decir sin un espacio en el lado derecho, pero esto son casos particulares, por lo tanto lo anterior es una definición general de las palabras.

Las palabras, al contrario de las oraciones, se puede decir que son limitadas en cualquier lenguaje, y normalmente una persona sólo utiliza una parte de todas ellas, por esto es razonable pensar que si las puede almacenar en su cabeza, al contrario de lo que se piensa sobre las oraciones, como se mencionó en la Sección 2.3.

Se suele clasificar las palabras en diferentes categorías léxicas. Esta clasificación puede hacerse a partir de diferentes criterios, como significado, forma morfológica y función sintáctica. Esta clasificación es importante conocerla, pues las reglas que rigen la forma en que se combinan u ordenan las palabras, según la gramática libre de contexto (el cual será el método implementado en este trabajo de tesis) no toman palabras en particular, sino categorías o tipos de palabras [22].

La categoría según su función sintáctica, conocida como categoría sintáctica, es la que define como puede actuar una palabra o que función puede cumplir, por ello es bastante útil a la hora de formar oraciones a partir de las reglas definidas por la sintaxis.

Algunas palabras pertenecen a más de una categoría, y dependiendo de su ubicación se dice que son de una u otra categoría en dicho caso. Cada una de las palabras que se utilizan para formar las oraciones pertenecen al menos a una categoría de las ocho que existen en lo que se llama "partes del discurso", estas ocho categorías son:

1. Sustantivo (Noun)
2. Pronombre (Pronoun)
3. Verbo (Verb)
4. Adjetivo (Adjective)
5. Adverbio (Adverb)
6. Preposición (Preposition)
7. Conjunción (Conjuction)
8. Interjección (Interjection)

Una de las unidades gramaticales del inglés son las palabras, pero además se tienen las frases, cláusulas y oraciones. Las palabras forman las frases, estas a su vez se unen para formar cláusulas y las cláusulas forman las oraciones. Una oración puede tener una o más cláusulas [24]. La Figura 2.2 muestra una representación gráfica de esto.

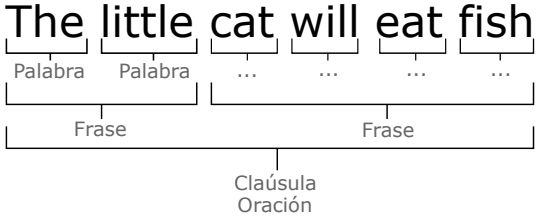


Figura 2.2: Componentes gramaticales.

Para formar oraciones, existen una serie de reglas que rigen la forma en que estas se generan. La sintaxis es la rama que se encarga de estudiar e identificar dichas reglas, para que a partir de los componentes más básicos de las unidades gramaticales se generen oraciones gramaticalmente correctas [22].

Estas reglas son la reglas léxicas, conocidas en inglés como *lexical rules*, reglas que indican qué palabras pueden ser usadas en los constituyentes los cuales a su vez pueden formar parte de otra frase (constituyente) [25].

En inglés existen diferentes tipos de oraciones, son cinco tipos y son declarativas (*statement*), interrogativas (*question*), imperativas (*imperative*) y exclamativas (*exclamation*). Cada tipo de oración tiene un uso principal. Declarativas son aquéllas que dan información, interrogativas son utilizadas para preguntar o pedir información, imperativas para dar órdenes y exclamativas para expresar sentimientos [24]. Sin embargo también pueden tener otros usos, no están restringidas a su uso principal.

Las oraciones de tipo declarativas, son oraciones que tienen una frase sustantiva (*Noun Phrase, NP*) seguido de una frase verbal (*Verbal Phrase, VP*) [11]. Estas oraciones, generalmente serán oraciones que brinden información, por lo tanto si lo que se busca, como es el caso de esta investigación, es extraer información para poder responder alguna pregunta, lo más indicado sería analizar este tipo de oraciones.

2.4.1. Determinador (Determiner)

Los determinadores (determiners) son palabras que aparecen al principio de las frases sustantivas (en inglés, *noun phrases*). Indican si la frase es específica o general. Por lo tanto se dice que los *determiners* son específicos o generales [26].

Ejemplo de "determiners" son "the" , "this" antes de un sustantivo. Como en las siguientes dos oraciones.

" *The* car was parked."

"*This* cute house is for sale."

2.4.2. Sustantivos (Noun)

Los sustantivos (*nouns*), desde el punto de vista funcional, son aquéllos que pueden ser precedidos de un determinador, muchos de ellos pueden tomar una forma en plural, pueden tomar posesivos entre otras cosas [11].

Son aquéllos que se utilizan para referirse al sujeto, objeto directo e indirecto de una oración. Los sustantivos pueden clasificarse en propios (*proper*) y comunes (*common*), los sustantivos propios en el inglés escrito aparecen con su primera letra en mayúscula. Otra

diferencia entre los propios y los comunes es que los primeros no están precedidos por un determinador. Para visualizar lo mencionado se pueden nombrar algunos sustantivos:

Propios:

- Tom
- Angela
- Andrew

Comunes:

- Book
- Water
- Cookie

Otro tipo de clasificación para los sustantivos es contables (*countable*) y no contables (*uncountable*). Los no contables poseen una sola forma al contrario de los contables, los cuales pueden tener una forma singular y una forma plural.

Lo más común para llevar un sustantivo común de singular a plural es agregarle una "s" al final de la palabra. Pero esta acción no es válida para todos los casos pues existen excepciones bien definidas por reglas de la gramática.

Pronombres (Pronouns)

Normalmente se usan en lugar de una frase sustantiva, o algún sustantivo. Por lo tanto muchas veces en una oración ocupan el lugar del sujeto u objeto, en especial los pronombres personales. Los pronombres no están precedidos por determinadores. Existen muchos pronombres, entre ellos *I*, *She*, *Him*, alguno de sus usos se puede entender observando las dos oraciones siguientes.

"*Maria* was eating pizza."

"*She* was eating pizza."

Estas dos oraciones tienen el mismo significado, la diferencia entre ellas es que la primera utiliza un sustantivo propio y la segunda sustituye este sustantivo por un pronombre.

2.4.3. Verbos (Verbs)

Los verbos tienen diferentes formas, estas son, una forma base, una s-forma o tercera persona del singular, forma de pasado, forma de gerundio y una forma de pasado participo. Por ejemplo, respectivamente para el verbo look se tiene:

- look
- looks
- looked
- looking
- looked

Estas diferentes formas de los verbos se usan dependiendo del tiempo verbal de la oración o del sujeto que realiza la acción. Sin este conocimiento es improbable formar una oración gramaticalmente correcta. Existen diferentes tiempos verbales en inglés, pero debido a que no es el objetivo principal de esta tesis se deja al lector investigar esto en caso de necesitarlo.

2.4.4. Preposición (Preposition)

Una preposición, es aquélla que generalmente ocurre antes de una frase sustantiva y complementa la oración.

"Julian is *at* home."

"This is *from* my grandmother."

En estas dos oraciones, se tienen dos preposiciones, las cuales son *at* y *from*. Como se ve, las preposiciones preceden el objeto de la oración.

2.4.5. Adjetivos (Adjectives)

Continuando con los tipos de palabras que conforman las diferentes partes del discurso, se tiene los adjetivos, en inglés se conocen como *adjectives*. Estos generalmente aparecen antes de los sustantivos y en algunas ocasiones aparecen luego del verbo.

Los adjetivos son palabras que califican el sustantivo, nos dan más información sobre ellos. Estos pueden ser palabras relacionadas a colores, texturas, tamaño entre muchas otras cosas. La siguiente lista muestra algunos ejemplos:

- Red

- Soft
- Easy
- Cute

2.4.6. Adverbios (Adverbs)

Los Adverbios en inglés *adverbs*, son usados para dar más información sobre el verbo, para describir de qué forma sucedió o sucede una acción, de qué manera está hecho o para indicar cuando se realizó algún evento. También pueden modificar una frase por completo.

Existen varias clasificaciones para los adverbios, entre ellas se tiene, adverbios de lugar, de cantidad o grado, de modo, de tiempo [11].

A diferencia de los adjetivos, un adverbio se puede encontrar en diferentes partes dentro de una oración, esto hace que sea más complicado definir una única regla de ubicación para este tipo de palabra [27]. Su ubicación depende en gran parte del tipo de adverbio, pero también puede variar dependiendo del verbo. En las siguientes dos oraciones se puede ver un pequeño ejemplo de ello:

"Mangoes *always* taste best when you pick them straight off the tree."

"My students are *always* on time."

En estas dos oraciones, se observa que *always* modifica al verbo respectivo, es decir es adverbio de modo, pero se puede observar que está en diferentes posiciones, en un caso uno está antes del verbo y en el otro caso después del verbo.

En este caso sólo estamos considerando al adverbio de modo, adverbio que modifica el verbo, pero también el adverbio puede modificar adjetivos, y otros adverbios, por lo que las diferentes posibilidades de dónde encontrarlo aumentan aún más.

2.4.7. Reglas

Existen diferentes reglas gramaticales, que ayudan a las personas que hablan cierta lengua a saber cómo conjugar un verbo, cómo cambiar un sustantivo contable (*countable noun*) de singular a plural, convertir algunos verbos en sustantivos y entre muchas otras cosas.

Por ejemplo, cuando se conjuga un verbo en presente con la tercera persona del singular en inglés, she/he/it se le aplica una pequeña modificación al mismo. Esta modificación consiste en agregarle "s" o "es" al final de verbo según su terminación, y en algunas ocasiones, se modifica el final de la palabra original además de agregarle alguna de las terminaciones mencionadas. Para pasar un sustantivo de singular a plural ocurre esto mismo.

Existen muchas otras reglas que generalmente pueden ser encontradas en el apéndice de un libro para aprender un idioma, si el lector desea saber un poco más de esto se puede referir a este tipo de libros.

2.5. Gramática libre de contexto

Como se ha mencionado anteriormente, gramática libre de contexto (CFG por sus siglas en inglés) se refiere a un conjunto de reglas que describen cómo se pueden ordenar y agrupar los símbolos, y un léxico de palabras y símbolos; es lo que los lingüista conocen como gramática generativa. Dichas reglas pueden ser definidas usando otras reglas definidas previamente [11].

Los símbolos que se utilizan para formar las reglas en la gramática libre de contexto, son denominados terminales y no terminales. Terminales son las diversas palabras que existen en inglés, es decir el léxico, como *cat*, *go*. No terminal se refiere a la agrupación o generalización de los terminales, con lo cual se tiene por ejemplo *noun* (N) o *adjective* (A).

Cada regla, tiene símbolos a la derecha o a la izquierda de la flecha. Los elementos a la derecha son uno o varios terminales o no terminales que aparecen de forma ordenada, y el elemento de la izquierda es un no terminal, que expresa una generalización, como en las ecuaciones 2.1 y 2.2.

$$\underbrace{NP}_{nonterminal} \rightarrow \underbrace{D}_{nonterminal} \quad \underbrace{N}_{nonterminal} \tag{2.1}$$

$$\underbrace{NP}_{nonterminal} \rightarrow \underbrace{The}_{terminal} \quad \underbrace{car}_{terminal} \tag{2.2}$$

Las letras de las ecuaciones 2.3 a 2.6, válidas para oraciones de tipo declarativas, representan lo siguiente:

- S : Oración (Sentence)
- NP : Frase sustantiva (Noun Phrase)
- VP : Frase verbal (Verbal Phrase)
- PP : Frase de preposición (Prepositional Phrase)
- D : Determinador (Determiner)
- A : Adjetivo (Adjective)
- N : Sustantivo (Noun)

- V : Verbo (Verb)

$$\mathcal{S} \rightarrow \mathcal{NP}\ \mathcal{VP} \tag{2.3}$$

$$\mathcal{NP} \rightarrow (\mathcal{D})\ (\mathcal{A})^*\ \mathcal{N}\ (\mathcal{PP}) \tag{2.4}$$

$$\mathcal{VP} \rightarrow \mathcal{V}\ (\mathcal{NP})\ (\mathcal{PP}) \tag{2.5}$$

$$\mathcal{PP} \rightarrow \mathcal{P}\ \mathcal{NP} \tag{2.6}$$

El asterisco en A, quiere decir que puede ser un número indefinido de adjetivos. Los paréntesis que aparecen en algunos constituyentes o tipos de palabras significan que son opcionales.

Para observar este tipo de reglas o estructuras de la oración se suele utilizar un árbol sintáctico como el de la Figura 2.3. Esto se hace con el fin de entender de forma prácticas las reglas.

Llevar una oración a la forma en que se observa en la Figura 2.3 se denomina parsing. Como se observa, lo que busca el parsing es ubicar cada palabra en tipo de frase (NP, VP, PP) en función de los tipos de palabras que se tienen. Es una forma visual de ver cómo se aplican las reglas a una oración.

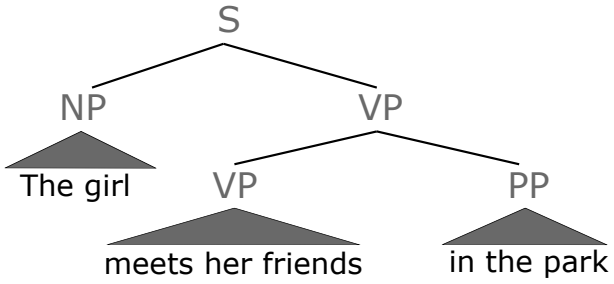


Figura 2.3: Árbol Sintáctico.

Capítulo 3

Método Propuesto

En este trabajo se analizan oraciones en inglés, aplicando procesamiento del lenguaje natural utilizando programación orientada a objetos con C++. Para lograr los objetivos planteados se implementaron diferentes clases que realizan cada proceso necesario para llevar a cabo cada una de las tareas. Además, se utilizó una base de datos con un gran número de palabras.

3.1. Método

Como se ha mencionado el lenguaje es un sistema con un conjunto de palabras que se unen para generar elementos más complejos que permiten transmitir información, dar órdenes y hacer preguntas. Algunos lingüistas lo estudian a través de reglas que permiten formar oraciones y a su vez el discurso. Tomando este enfoque de la sintaxis, reglas, se puede decir que como lo humanos usan dichas reglas para comunicarse, se extrapola esta idea a las computadoras, para hacerlas entender como los humanos entienden el lenguaje natural.

Si se conocen y estudian las reglas, estas se pueden llevar a un lenguaje de programación y de esa forma hacer entender a la máquina lo que un humano entendería.

Retomando lo mencionado en el Capítulo 2, la sintaxis estudia las reglas que se utilizan para formar una oración, por ello la sintaxis es una herramienta que se va a utilizar para abordar la PLN desde esta perspectiva de reglas. Estas reglas que rigen cómo formar las oraciones, no hacen referencia a palabras en particular, sino más bien a categorías o tipos de palabras [21]. Debido a esto, es necesario clasificar de alguna forma las palabras de acuerdo a su posible función sintáctica.

En este trabajo se parte del enfoque racionalista apoyado por Noam Chomsky en su libro estructura lógica de la teoría lingüística, a partir de donde sale el término de gramática generativa.

Lo que se pretende es darle todo el conocimiento previo posible y mecanismos de razonamiento a la máquina, en este caso reglas y condiciones para replicar lo que se cree que

pasa en la mente de los seres humanos, según la lingüística generativa, a la hora generar oraciones. En la gramática generativa, se habla que los seres humanos poseen un léxico, el cual alojan en su memoria y lo utilizan para formar las oraciones de acuerdo a las reglas. Por ende la máquina debe tener también un conocimiento previo de palabras, para ello se utiliza una base de datos que agrupa las palabras según su categoría léxica, aunque algunas veces puede ser que una palabra aparezca en más de un archivo de la base de datos, pues existen palabras que pueden pertenecer a más de una categoría léxica.

Para la programación se implementaron varias clases, esto con la finalidad de agrupar las funciones y variables que conciernen a cada uno de los diferentes procesos que se ejecutan. Cada clase pretende abarcar diferentes tareas y variables que tienen relación entre sí. Las clases se crearon con base en los componentes esenciales de los algoritmos de PLN descritos en la literatura, la gramática libre de contexto y las reglas que rigen la derivación léxica de algunas palabras.

Como se ha dicho, se quiere que el algoritmo funcione como se piensa que funciona la mente de los humanos cuando se quiere utilizar el lenguaje natural. Por ello las clases implementadas en el algoritmo desarrollado también toman en cuenta esto. Por ejemplo, para la parte del "razonamiento" se utilizan varias clases, estas clases son "Lemmatization", "VerbTenses" y "ParsingCompiler". En referencia al léxico que posee el ser humano se crearon las clases "TokenDef" y "LexicalA" para procesar en conjunto con la base de datos las palabras de las oraciones.

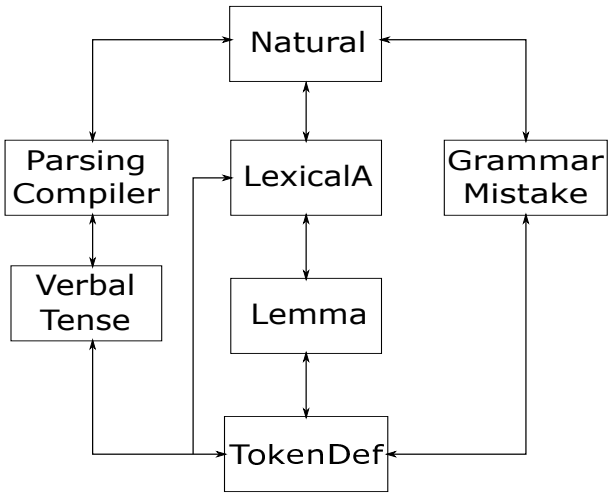


Figura 3.1: Clases del algoritmo y su interacción.

En la Figura 3.1, se pueden ver las interacciones que existen entre dichas clases. Natural es el programa principal el cual tiene acceso a todas las clases. ParsingCompiler tiene acceso a la clase VerbalTense, la cual a su vez tiene acceso a TokenDef, por lo tanto la primera tiene acceso a estas dos clases. GrammarMistake tiene acceso a TokenDef. En la figura la clase Lemma hace referencia a la clase Lemmatization.

Como se mencionó en el Capítulo 2, existen varios tipos de oraciones. En este trabajo se analizan oraciones del tipo declarativas, pues es a partir de este tipo de oraciones de donde se obtiene mayor información y se pueden contestar las preguntas ¿quién?, ¿qué? y posiblemente el ¿cuándo?. Se dice que posiblemente pues la respuesta al cuando no está siempre presente en las oraciones.

3.2. Standard Template Library

Para los algoritmos desarrollados se utiliza la librería STL (Standard Template Libary). Esta librería fue desarrollada por Alexander Stepanov and Meng Lee. STL es una librería de componentes, estos componentes son los contenedores, algoritmos e iteradores [28].

Para este trabajo se usó set, vector, wstring (contenedores) e iterator. Para manejar la entrada de datos, los cuales son cadenas de caracteres, se maneja el tipo de variable wstring, el cual encapsula datos tipo wchar_t.

Para tener la base de datos cargada en el algoritmo se utiliza set. Este componente, en este caso, almacena cadenas de caracteres en formato wstring, a las cuales se hace referencia por ahora como palabras. Las palabras que son almacenadas en los diferentes set provienen de los datos que se extraen en los diferentes archivos de texto de la base de datos, a través del algoritmo implementado para tal fin.

Vector es otro contenedor utilizado en los algoritmos, este funciona parecido a un arreglo C++. Se seleccionó un vector en lugar de un arreglo, pues el primero tiene un tamaño dinámico, lo que es bastante útil en este caso, dado que por ejemplo, en el caso de la oración a evaluar, no se sabe a priori la cantidad de palabras que contendrá. En este trabajo se utiliza el tipo de variable vector para almacenar objetos de una clase creada para este proyecto llamada TokenDef, la cual puede almacenar datos de tipo entero, wstring y bool.

Iterator es utilizado repetidas veces para recorrer la cadena de caracteres y vectores con el fin de cumplir el objetivo de ciertas funciones esenciales del algoritmo. Un iterador cumple la misma función que un puntero, por lo tanto se puede entender y manejar como tal.

3.3. Base de datos (léxico)

El objetivo principal de este trabajo, es analizar oraciones, y puesto que las oraciones están compuestas por palabras, la máquina de alguna forma debe conocer un número considerable de las mismas. Lo ideal sería conocer un número extremadamente grande de palabras, pues mientras más conocimientos mayor exactitud o mejor desempeño se puede tener, pero esto no es necesario.

Los mismos humanos, quienes son los que utilizan el lenguaje natural, no tienen un conocimiento infinito de palabras. Con el conocimiento de una porción del vocabulario total existente, estos pueden crear infinidad de oraciones diferentes, con las pocas palabras que tengan en su léxico y realizando derivaciones léxicas de las palabras conocidas.

En este sentido, se creó una base de datos de palabras para crear el léxico del algoritmo. Estas palabras se agrupan en diferentes archivos de texto (.txt) con codificación Unicode. Se utilizó este tipo de formato pues incluye los caracteres de uso común y en la actualidad se utiliza en mayor medida que el ANSI. Existen varios archivos de texto porque las palabras son agrupadas de acuerdo al tipo de palabra. Se manejaron 21 diferentes tipos de palabra.

Los grupos son clasificaciones básicas de la gramática, así como unas cuantas subcategorías (categorías de clases principales). Los diferentes tipos de clasificaciones se eligieron de acuerdo a las reglas que rigen la estructura de una oración, los diferentes tiempos verbales y con base en las reglas que se utilizan para llevar una palabra de su base (lema) a una forma flexionada, como plural, tercera persona del singular y gerundio.

La base de datos fue creada con diferentes fuentes de palabras, como conocimiento propio del autor, recursos de web y algunos artículos [11].

3.4. Clases

Las clases presentes en el algoritmo son las siguientes:

- TokenDef
- LexicalA
- Lemmatization
- VerbTenses
- ParsingCompiler
- GrammarMistake

La primera es para definir las características (variables) del "token" (componente léxico) que serán utilizadas durante todo el análisis de las oraciones. La clase "LexicalA"se encarga de clasificar cada una de las palabra que conforman la oración ingresada por el usuario.

"Lemmatization" ayuda a identificar palabras que no están es su forma base y las clasifica en alguna de los posibles tipos de palabras. "VerbalTenses", identifica si el tiempo verbal de la oración es válido y la última es la que contiene las reglas principales que rigen la forma en la cual se forma una oración.

"ParsingCompiler" es la clase que posee las reglas de la gramática libre de contexto.

Finalmente "GrammarMistake", es la clase utilizada para buscar algún error gramatical definido en el algoritmo.

3.4.1. Definición de Token (TokenDef)

Para empezar se debe entender como se emplea el término *token* y qué representa en este trabajo. Un "token", es un carácter o conjunto de caracteres que tiene grandes probabilidades de representar una palabra, un número y en ocasiones signos de puntuación.

En el algoritmo, un "token" es un objeto que tiene varias características o variables. En la Figura 3.2 se observa que se tienen cinco variables. La variable *word*, se utiliza para guardar la cadena de caracteres que representa dicho *token*, la variable *type* se utiliza para asignar el tipo o clasificación de la palabra, estas dos variables son esenciales. Las demás variables, *line*, *numword* y *startUpper* son variables que se utilizan como herramientas de apoyo para los algoritmos.

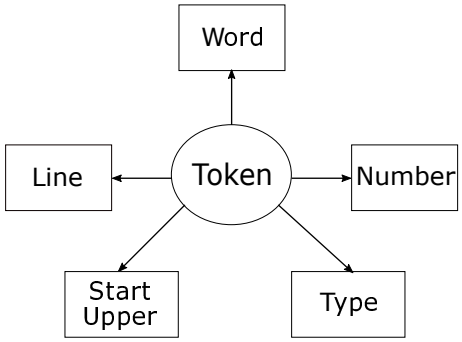


Figura 3.2: Objeto *token*. En la figura se pueden apreciar sus diferentes variables.

En esta clase se crearon variables para las diferentes clasificaciones de los *tokens*. Cada clasificación tiene un número entero único, que hace referencia a una posición de bit particular. Como los números se refieren a una posición de bit, todas las variables definidas para los *tokens* representan un número entero en base dos. Para el algoritmo se definieron 27 tipos de *tokens*, por lo que la primera variable representa el valor de 0 y la última representa el valor de 2^{27}.

En esta clase contiene las funciones que son capaces de cargar los diferentes archivos de texto de la base de datos en los diferentes contenedores de tipo *set* (STL). Por ser un contenedor de tipo *set*, el cual almacena elementos únicos de forma ordenada, no existirán palabras duplicadas en el mismo. Todas las palabras de la base de datos se guardan en los contenedores con todas sus letras en minúscula, esto es parte del procesamiento previo del texto.

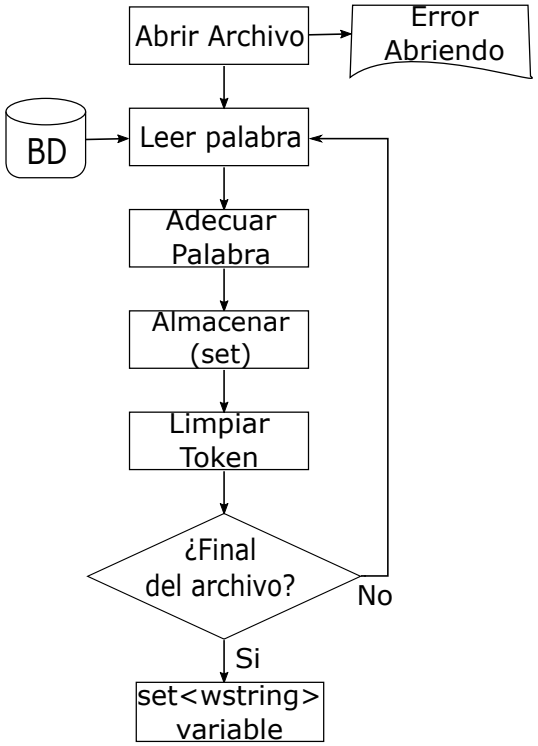


Figura 3.3: Diagrama de la clase "TokenDef".

El diagrama de la Figura 3.3, muestra cómo se desarrolla esta clase. Se puede ver cómo se obtienen dos salidas de esta clase, una es un mensaje de error al tratar de abrir alguno de los archivos de la base de datos y no lograr dicho objetivo y la otra, la esperada, es un *set* de tipo *wstring* que contiene las palabras de la base de datos en este tipo de contenedor, una en cada posición y todas con minúsculas.

Para esta clase se tienen las variables y métodos que se muestran en el digrama UML de la Figura 3.4.

3.4.2. Lematización (Lemmatization)

Esta clase se crea con la finalidad de hallar el lema de una palabra dada que no se encuentre en la base de datos.

El lema de una palabra se refiere a la forma base de dicha palabra, dicha palabra puede haber cambiado ya sea para convertirse en un gerundio o un verbo conjugado en la tercera persona del singular, por nombrar algunos ejemplos.

Dicha clase permite o contribuye a disminuir la capacidad de almacenamiento requerida por la base de datos. Dado que con esta clase se expande el vocabulario que es capaz de manejar el algoritmo sin necesidad de incluir más palabras en la base datos, por medio de las reglas existentes para formar otras palabras a partir de palabras básicas que existen en

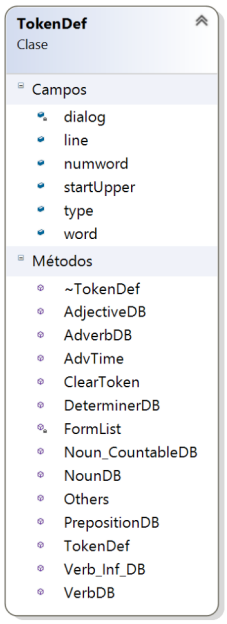


Figura 3.4: UML Clase "TokenDef".

la base de datos. Por ejemplo, un gerundio y un verbo conjugado para la tercera persona del singular se pueden formar a partir de un verbo en su forma infinitiva sin "to". El plural de un sustantivo contable se puede generar conociendo el sustantivo en singular y agregándole "s" o "es" al final según sea el caso particular.

Para lograr el objetivo, se lleva la palabra ingresada por el usuario a su posible lema y se ubica dicha palabra en la base de datos. Si se logra, dicha palabra adquiere la clasificación que le corresponda, para posteriormente hacer el análisis de la oración completa. Los métodos y variables de esta clase se muestran en el diagrama UML de la Figura 3.5.

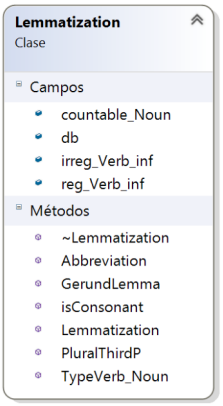


Figura 3.5: UML de la clase "Lemmatization".

En esta clase existen dos funciones principales las cuales se llaman "GerundLemma" y

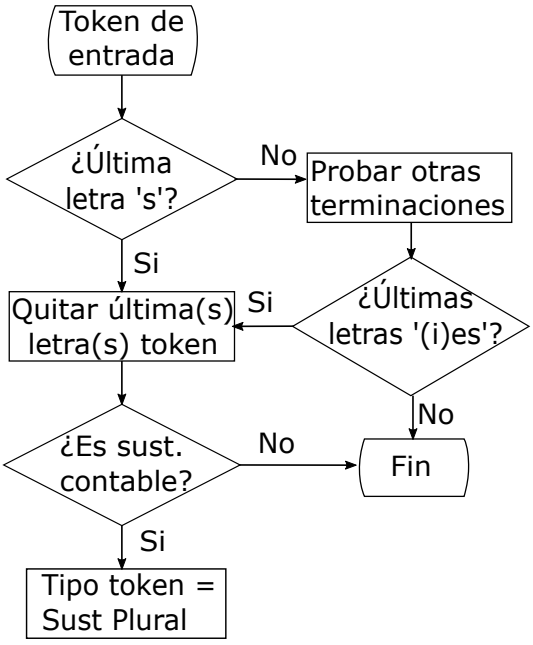


Figura 3.6: Clase "Lemmatization", clasificación sustantivos plurales.

"PluralThirdP". La primera se utiliza para determinar si un token ingresado por el usuario, sin clasificación hasta el momento, es un gerundio. Para ello se verifica si la palabra termina en "ing" (terminación obligatoria de todos los gerundios). Si el token termina en ing, se le quita estas tres últimas letras y se intenta ubicar la palabra en los sets que corresponden a los verbos. Si no se consigue, se intenta modificando un poco la terminación de la palabra sin el ing de acuerdo a ciertas reglas gramaticales que se aplican para llevar determinadas palabras a la forma de gerundio.

Tales reglas son como la que se aplica en la siguiente palabra:

"Slide"

Como este verbo termina en "e", se remueve esta letra y se le agrega ing a la palabras para llevarla a la forma de gerundio. Esta regla se aplica para todas las palabras con dicha terminación.

Si no se toma en cuenta esto en el algoritmo, al tener una palabra como "sliding", se le quitaría la terminación "ing"quedando "slid", al tratar de buscar esta palabra en el vocabulario, no se encontraría, lo que causaría que la palabra siga sin clasificarse. Por lo que se agregó esta y otras reglas más para poder manejar este tipo de casos. En este algoritmo se aplica a la inversa la regla gramatical, pues se busca ir del gerundio al verbo y no al revés.

La segunda función mencionada ("PluralThirdP"), Figura 3.6, se utiliza para determinar si una palabra desconocida pertenece al tipo de sustantivos contables o de verbos. En el

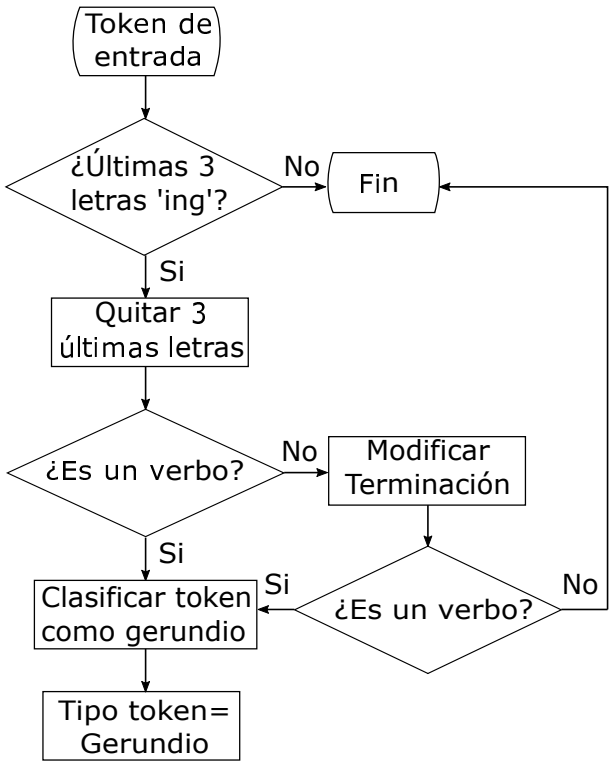


Figura 3.7: Clase "Lemmatization", pasos para clasificar un gerundio.

primera caso, quiere decir que el sustantivo es un sustantivo en plural, y el segundo caso, que se conjugó el verbo para la tercera persona del singular. Estos dos casos se agrupan pues las reglas gramaticales son las mismas. Al igual que los gerundios, existen ciertas reglas para algunas terminaciones en particular, estas también se toman en cuenta en este caso. La base de la palabra, sin la transformación a plural o verbo conjugado, se busca en los set de sustantivos contables o de verbos, dependiendo de si se encuentra y en qué set se encuentre se le asigna la clasificación.

En el diagrama de la Figura 3.7, se observa el proceso mencionado para determinar si una palabra es gerundio. También se puede apreciar que la salida es la clasificación del token en gerundio en caso de cumplir con la reglas para pertenecer a tal clase, en caso contrario no se obtiene ninguna salida. De la segunda función, se obtiene una posible clasificación del token como sustantivo plural, como se observa en la Figura 3.6.

3.4.3. Analizador Lexicográfico (LexicalA)

El analizador lexicográfico, es llamado "LexicalA" en el algoritmo desarrollado para esta tesis. Sus tareas principales consisten en dividir la oración en palabras, números, signos y clasificar las diferentes palabras presentes en la oración ingresada por el usuario. Esta clase se basa en la tokenización. Su funcionamiento está basado en el diagrama de la Figura 3.8, y para hacer posible dicho funcionamiento se crearon las variables y funciones del digrama UML mostrado en la Figura 3.9.

Cuando un token no se puede clasificar al momento de compararlo con los diferentes set de la base de datos, antes de etiquetarlo como desconocido, se hace uso de la clase Lemmatization.

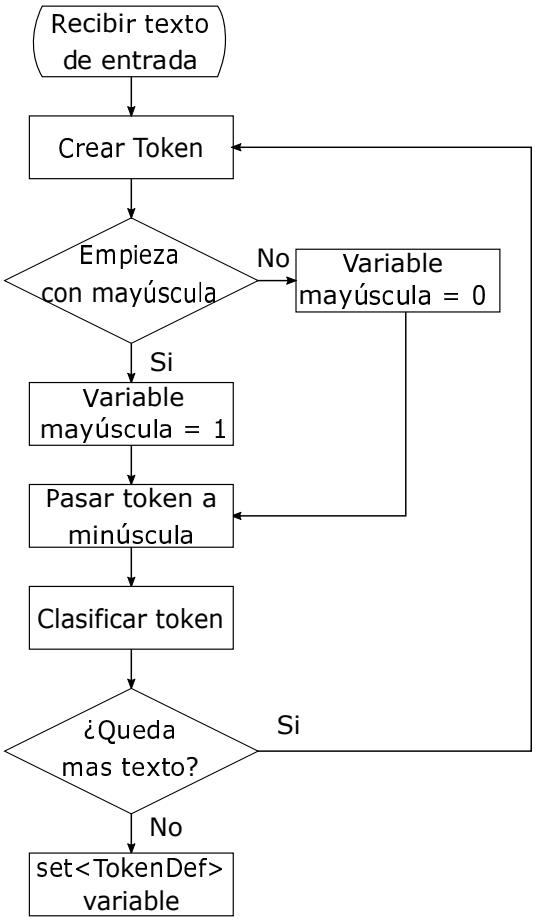


Figura 3.8: Diagrama del analizador lexicográfico.

Al crear un objeto de esta clase, se carga la base de datos del vocabulario en los diferentes contenedores de tipo set, definidos en la clase, que alojan valores de tipo wstring, los cuales no son más que las palabras de dicha base de datos. Para poder realizar esta tarea, esta clase utiliza los metodos de la clase TokenDef definidos para tal fin.

Tokenización

El principal propósito de esta tarea consiste en tokenizar los diferentes caracteres que ingresa el usuario al escribir la oración que se desea analizar.

En el análisis léxico, la tokenización consiste en agrupar una cadena de caracteres en palabras, signos de puntuación, números o cualquier expresión que tenga un significado coherente

dentro del lenguaje que se está utilizando, en este caso, el inglés [29]. Esta es la primera acción que se realiza en la clase LexicalA, dividir y almacenar cada palabra de la oración en un vector.

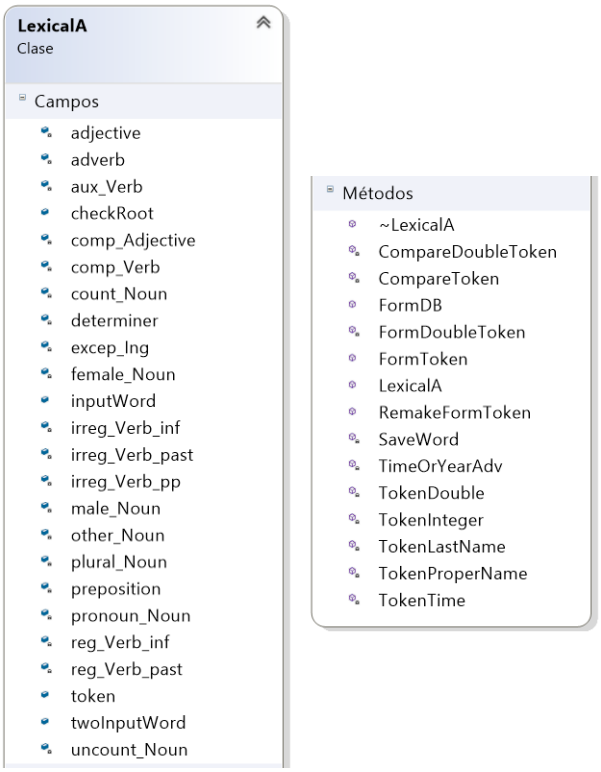


Figura 3.9: Diagrama UML clase "LexicalA".

Cuando se ingresa una oración, esta clase entra en juego. Diferentes funciones de la clase, se encargan de identificar el inicio y fin de un token, clasificarlo, asignarle un número de palabra y línea, determinar si empieza o no con mayúscula, pasar el texto del token a minúsculas y finalmente guardar dicho token en un set, en este caso llamado inputWord.

La clase también es capaz de procesar e identificar un token que esté constituido por más de una palabra, como es el caso de los verbos conocidos como "Phrasal Verbs" en forma no separable, los cuales tienen más de una palabra. Como por ejemplo:

"Wake up","Take off"

Estas dos palabras en conjunto representan un verbo. Cuando esto ocurre se tiene un token de dos palabras con su clasificación respectiva como verbo.

Etiquetado (Tagging)

El etiquetado, como se puede inferir, consiste en poner etiquetas, es decir clasificar algo, en este caso tokens. Este proceso se realiza luego de la tokenización, los diferentes tokens

que se encontraron son comparados con las palabras de la base de datos, y según si existe o no coincidencia, se le asigna una, varias o la categoría “desconocido”.

Como se mencionó en la Sección 2.4, las palabras pueden pertenecer a más de un grupo. Por ello a la hora de aplicar una clasificación se debe considerar esto, por eso se menciona que se le puede asignar varias categorías léxicas a una sola palabra. Para hacer esta consideración el analizador léxico, asigna las diversas clasificaciones que puede tener una palabra de manera simultánea. Posteriormente será tarea de otras funciones, decidir qué hacer con esta clasificación y cuál tomar en cuenta a la hora de analizar la oración completa.

Para poder permitirle a un token esta posibilidad, la variable de tipo se maneja como una serie de bits consecutivos donde cada posición de un bit representa una categoría en partículas. En caso de que el bit de la posición n esté en 1, significa que la palabra pertenece a la clase que dicho bit en la posición n representa (según se definió en los macros de la clase TokenDef). Como cada bit en la posición n representa una sola y única categoría, esta variable, vista de esta forma, da la posibilidad de ubicar una palabra en diferentes tipos de categorías y posteriormente el algoritmo puede procesar esta información sin ningún problema.

Para entender un poco mejor el proceso se puede observar la Figura 3.10. En esta figura se tiene un número de 9 bits, en donde 4 bits están en 1. Si una palabra tuviese esta valor en la variable tipo, pertenecería a 4 categorías, las cuales serían las que el bit en la posición 0, 2, 3 y 6 representa. En el algoritmo cada macro definido para cada categoría no es más que una máscara, la cual permite extrar el valor de cierto bit de la variable tipo con la ayuda de la operación “and”.

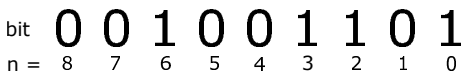


Figura 3.10: Variable tipo, representa la o las categorías a las que pertenece una palabra.

Para observar esta situación en la que una palabra puede pertenecer a más de una categoría (por ejemplo dos categorías) piense en la palabra “content”. Esta palabra puede referirse al estado de sentirse contento, lo que hace que la palabra sea un adjetivo, pero también puede referirse a “contenido” lo que la convierte en un posible sustantivo. Por lo tanto esta palabra tendrá el menos dos etiquetas a la hora de clasificarla.

Por el contrario, en lugar de tener varias categorías, una palabra puede quedar sin categoría léxica, es decir, queda clasificada como desconocida. Debido a esta posible clasificación del token como desconocido, surgen problemas a la hora de querer reconocer algún tipo de frase que incluya dicha palabra, pues al ser desconocida, las reglas no se cumplirán pues ninguna regla incluye una categoría desconocida. Así, cada vez que una palabra sea desconocida se debe tomar alguna acción de ser posible. Se habla de acciones pues por muy grande que sea la base de datos y las reglas de lematización, es casi imposible tener todas las palabras existentes en el idioma, ¿conoce usted todas las palabras de su lengua?.

Para contrarrestar el efecto negativo que generan las palabras desconocidas en una oración se deben tomar ciertas acciones. A continuación se mencionan estas acciones.

En las oraciones aparecen nombres propios junto con su apellido, puesto que no hay ninguna base de datos que se refiera a apellidos, esta palabra se clasificaría como desconocida. Se podría tener una gran base de datos para los nombres pues estos pueden considerarse finitos, pero el autor considera que con los apellidos no se puede hacer lo mismo, pues existen infinidad de apellidos y es mas difícil considerarlos finitos pues se limitaría mucho los nombres disponibles para el análisis.

Para evitar que quede como palabra desconocida un apellido, se asume que cualquier palabra desconocida que empiece con mayúscula y que se encuentre después de un nombre propio será un apellido, esto es solo una aproximación para realizar un intento de clasificación.

También pueden existir tokens de tipo numérico o signos de puntuación. Para este tipo de tokens no existen bases de datos, estos son clasificados de acuerdo a ciertas instrucciones dentro de la clase.

Algunas veces no se encuentra coincidencias de los tokens con las palabras de la base de datos, esto puede ser debido a que el mismo no aparece en su forma base. Si este es el caso, la palabra aún puede ser clasificada puesto que "LexicalA", utiliza la clase "Lematization"para manipular los tokens que al momento son desconocidos y que podrían ser una derivación de una palabra existente en la base de datos.

Lo que se busca con estas acciones es aumentar de alguna forma el vocabulario que puede aceptar el algoritmo sin afectar su capacidad de procesar las oraciones y sin la necesidad de tener que agregar más palabras o clasificaciones a la base de datos.

3.4.4. Parsing (ParsingCompiler)

Esta clase se encarga de ubicar en las oraciones los diferentes tipos de frases definidos en la gramática libre de contexto, Sección 2.5. Para ello, el algoritmo con base en las reglas definidas anteriormente, ubica las diferentes frases sustantivas, de preposición y la frase verbal.

El diagrama de esta clase, de forma resumida, se puede observar en la Figura 3.11. En este primer algoritmo general, no se muestra la frase de preposición pues esta frase está dentro de la frase sustantiva y verbal, por lo que en los diagramas de más adelante, donde se visualiza con mayor detalle el funcionamiento, se podrá observar que aparece esta frase.

La entrada de este algoritmo es un vector que contiene variables de tipo tokenDef. Este vector se va recorriendo de uno en uno a medida que se van encontrando coincidencias de la variable con alguna de las reglas definidas. Por ejemplo, se empieza con la variable de la posición cero del vector, si esta palabra es un adverbio, se continúa evaluando el algoritmo con la variable de la posición uno y así sucesivamente.

Se observa además, que se busca una posible coincidencia del adverbio al inicio y al final del proceso, esto, con el fin de hallar un adverbio de tiempo que responda a la pregunta ¿Cuándo?. Este adverbio es opcional pues una oración no necesariamente debe tener un adverbio, caso contrario para la frase sustantiva y la frase verbal en una oración de tipo enunciativa; las cuales deben tener obligatoriamente un sustantivo y un verbo. Es por esta razón, que si no se consigue una frase sustantiva y una verbal, en ambos casos se finaliza el proceso, como se muestra en el diagrama, pues la oración ingresada no está cumpliendo con los requerimientos mínimos.

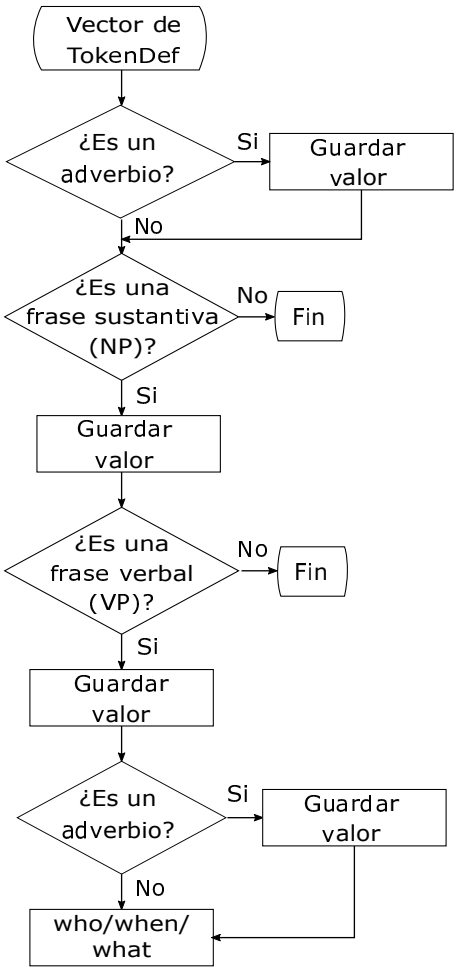


Figura 3.11: Diagrama general de la clase ParsingCompiler.

La frase sustantiva y verbal se guarda para responder las preguntas. Al terminar de ejecutarse el algoritmo, la salida será la respuesta a las tres preguntas, ¿Qué?, ¿Quién? y ¿Cuándo?.

El algoritmo permite analizar una entrada que esté compuesta por dos oraciones unidas por la conjunción “and”. Si se encuentra esta palabra al final de la primera oración, el proceso se vuelve a ejecutar. Este proceso se ejecuta indefinidamente hasta terminar de

agrupar todas las palabras en frases y a su vez en oraciones, siempre y cuando se encuentra el conector y los siguientes tokens realmente formen frases y oración(es).

Como se ha mencionado, las oraciones pueden tener o no adverbios, estos adverbios pueden aparecer en diferentes posiciones de la oración, pero los adverbios de tiempo aparecen al principio o al final de una oración, aunque esta ubicación no es exclusiva de los adverbios de tiempo, también se pueden encontrar de otro tipo y además puede ser más de uno. Como pueden existir dos adverbios seguidos, se toma en cuenta esto en el proceso de ubicación de un adverbio. Para dar un ejemplo, si se tiene la frase:

"John thumbed through the book *very rapidly*."

En este caso tendremos "very" y "rapidly", dos adverbios que aparecen juntos, con el proceso definido en el diagrama de Figura 3.12 se puede manejar este tipo de casos.

En el diagrama Figura 3.12, se puede ver que se tiene un índice, esta variable se utiliza a lo largo de todo el proceso, es la variable que permite llevar el seguimiento de la variable tokenDef del vector que se está evaluando en determinado momento. Con este índice se sabe qué posición del vector se debe evaluar. En el instante en que se tiene la entrada del usuario, se define el índice máximo al que se puede llegar, con esto se evita intentar acceder a posiciones de memoria del vector que no existen. Índice es una variable global que utilizan todas las funciones de la clase, cada una de ellas puede manipularla.

En la Figura 3.12 se observa que el índice se recorre, esto pasa en todas las funciones cuando se encuentra una coincidencia, pues como se mencionó, se evalúa una sola variable a la vez, del vector que contiene todas las palabras de la entrada del usuario. Para fines de practicidad no se muestra en los demás diagramas.

Cuando se desea ubicar la frase sustantiva, el proceso no es tan simple como buscar un sustantivo y definir dicho sustantivo como la frase sustantiva; hay otros detalles que se deben tomar en cuenta para esta tarea. En la Figura 3.13 se puede ver a grandes rasgos que el proceso es más complejo que sólo hallar un sustantivo. Se puede ver que si se consigue un pronombre se debe verificar que realmente este sea un pronombre y no un determinador, como puede ser el caso de "this", el cual en una oración como "this is my father"funciona como la frase sustantiva (en este caso la frase es sólo el pronombre), pero en el caso de la oración "this year can be the year of our success"funciona como un determinador pues está definiendo o determinando un año en particular.

En caso de que una palabra se clasifique como pronombre y luego al evaluar la palabra siguiente se determina que la anterior no puede ser pronombre pues no tendría sentido la oración, el algoritmo se regresa a la posición anterior del vector y sigue ejecutándose desde dicha posición.

En el diagrama de la Figura 3.13, se observa que también se busca hacer coincidencia con un adjetivo en una de las ramas del diagrama, este es una variable opcional que puede

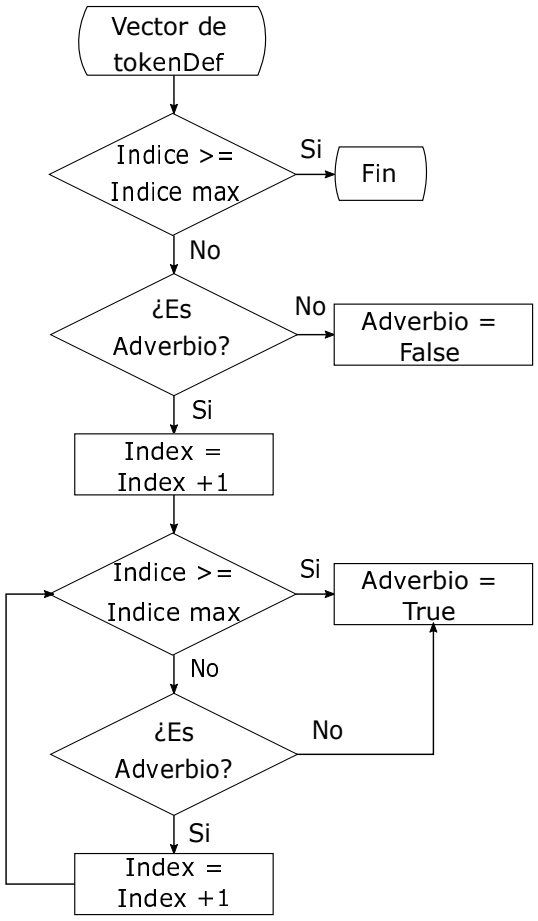


Figura 3.12: Proceso para encontrar posibles adverbios en la oración.

existir y en caso de que exista se debe ubicar para poder hacer coincidencia con las reglas de la gramática libre de contexto.

Los diferentes caminos que se pueden seguir en el diagrama, Figura 3.13, se originan por las diferentes reglas de la gramática libre de contexto y reglas gramaticales, muchas de las cuales se mencionaron en Subsección 2.4.2 y se menciona de forma un poco más extensa en la Sección 3.5.

Como se mencionó en la Sub-Subsección 3.4.3 (Etiquetado), existe una clasificación simultánea. Esta clasificación simultánea es necesaria, pero también trae ciertos problemas. Un ejemplo de ello es una palabra que sea simultáneamente sustantivo y adjetivo. Como se definió la regla de frase sustantiva, NP –> (D) (A)* N , en ese orden, se debe estar seguro entonces que la palabra es un A en la oración y no un N, de otra forma se puede clasificar el sustantivo como adjetivo y por ende no se va a cumplir la regla y no existirá una frase sustantiva, pues para que exista debe haber un sustantivo.

Para solventar este problema, cuando se analiza si la palabra es un sustantivo, se busca en la posición actual, si no es sustantivo, entonces se regresa a la posición anterior y verifica

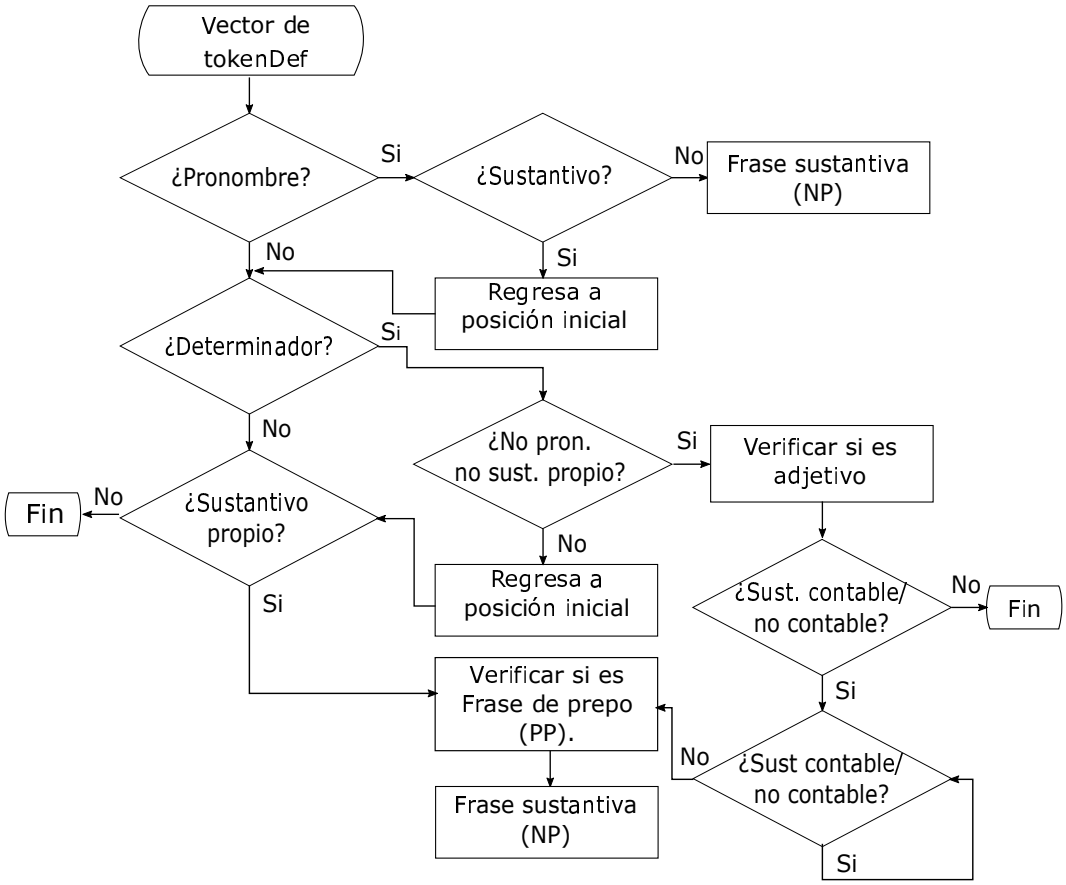


Figura 3.13: Proceso para encontrar una frase sustantiva (Noun Phrase).

si es sustantivo, si lo es, quiere decir que anteriormente se había tomado esa misma palabra como adjetivo cuando realmente estaba actuando como sustantivo en dicha frase. De esta forma se evita este posible error y se pasan las condiciones para lograr clasificar la frase como sustantiva.

De acuerdo a la producción 2.4, el NP está formado por un solo sustantivo, pero se observa que esto no siempre es así. Existen los sustantivos modificadores, que son sustantivos que aparecen antes de otro y muestran que una cosa es parte de otra. Un ejemplo de esto sería la siguiente cláusula:

"The old newspaper seller."

En esta oración "newspaper" y "seller" son ambos sustantivos, pero uno es parte del otro, en este caso newspaper es un sustantivo modificador. Esto debe ser tomado en cuenta a la hora de aplicar la producción mencionada para que sea correctamente analizada. Para ello se acepta que el sustantivo sea uno o más palabras de este tipo en el lugar donde corresponde que aparezca.

La frase verbal, como se describió en Sección 2.5, es de la forma VP ->V (NP) (PP), donde la V representa el verbo y debe existir siempre en la frase para que pueda ser clasificada como frase verbal. Los demás componentes no son obligatorios. En la Figura 3.14, en el primer punto de divergencia se pregunta si es un verbo, para esta consulta no sólo se busca un solo verbo, puede ser más de uno, debido a que existen diferentes tiempos verbales los cuales pueden tener varios verbos uno seguido de otro, quienes en conjunto forman un tiempo verbal particular. Para evaluar esto, en este punto el algoritmo se complementa con la clase "verbTense", la cual determina si el verbo o el grupo de verbos en conjunto pertenece a algún tiempo verbal válido.

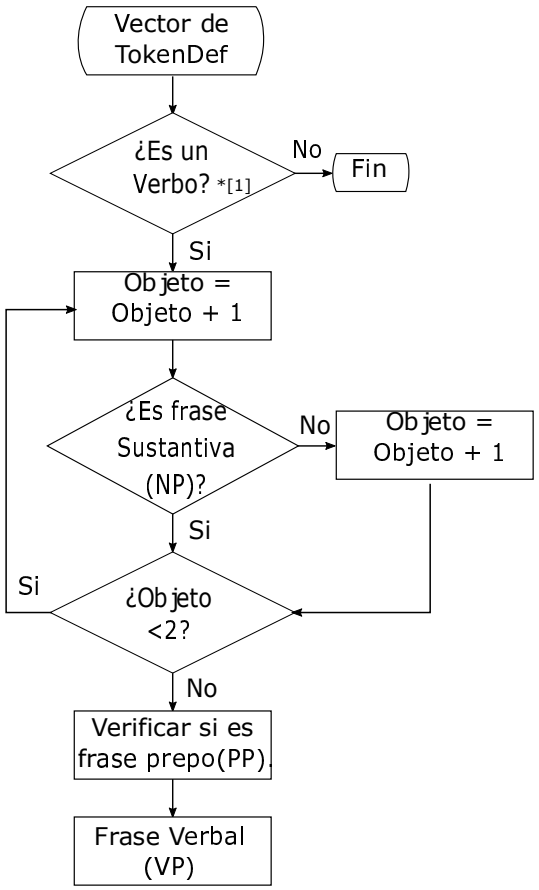


Figura 3.14: Diagrama descriptivo que define la forma de hallar la frase verbal en la oración. *[1] No sólo se hace coincidir un verbo, puede ser una serie de verbos que forman un tiempo verbal en particular.

Como el verbo es un componente esencial en esta frase, en el diagrama se ve que si no se tiene uno, no se sigue con el proceso pues ya se descarta que sea una frase verbal. Al ubicar este verbo o verbos, se ubica el ¿Qué? de la oración, pues es la acción que se está ejecutando por el sujeto. Un verbo puede o no tener objeto directo e indirecto, los objetos no son más que frases sustantivas, esto es lo que se verifica en los siguiente pasos del proceso, utilizando el proceso definido para hallar frases sustantivas. Antes de finalizar, se verifica si existe o no

una frase preposicional, para lo cual se tiene otro diagrama que describe este proceso.

La frase de proposición es otro conjunto de palabras que se va a encontrar con frecuencia pues puede estar presente en la frase sustantiva y en la verbal. En Figura 3.15, se puede observar en el diagrama que se busca la coincidencia con una frase sustantiva además de la coincidencia con la preposición, pues esta frase tiene dentro de ella misma la NP. En el proceso, primero se busca una coincidencia con un pronombre, si existe dicha coincidencia se busca una frase sustantiva, para lo cual se hace el mismo proceso que el definido anteriormente para manejar dichas frases, el cual a su vez puede llamar de nuevo a este mismo proceso de frase preposicional y el cual puede volver a llamar el proceso de hallar una frase sustantiva, teniendo así una importante recurrencia en el programa, lo cual hace esencial saber en qué parte del programa se está en determinado momento, o más preciso, cuál función inició el proceso. ¿NP invocó a PP o al revés?.

Esta es una de las clases más complejas y la que posee el algoritmo más largo entre todas las clases de este proyecto como se ha visto a lo largo de todo lo aquí descrito. En la Figura 3.16, se puede observar la estructura de esta clase estudiando el diagrama UML que allí se muestra.

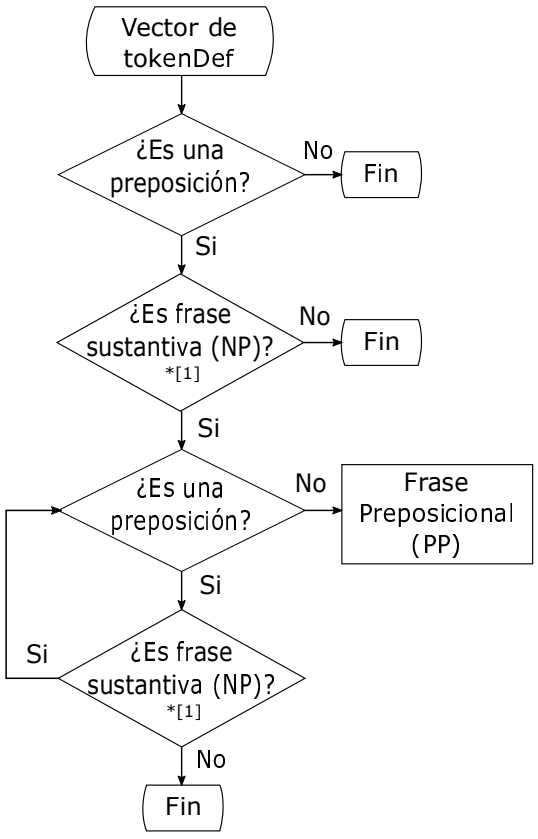


Figura 3.15: Este diagrama describe el proceso para hallar una frase preposicional, *[1] para ubicar la frase sustantiva se hace el mismo proceso que en la Figura 3.13.

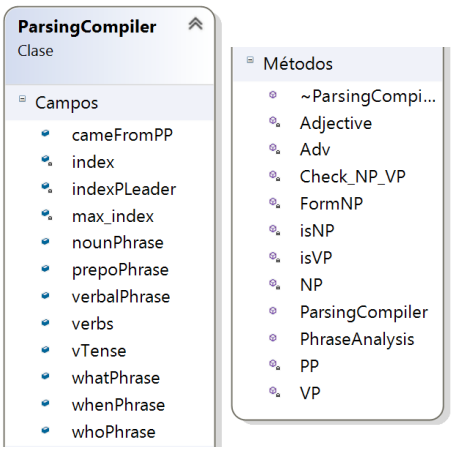


Figura 3.16: Diagrama UML de la clase "ParsingCompiler".

3.4.5. Error Gramatical (GrammarMistake)

Esta clase se desarrolló con la finalidad de encontrar algún error gramatical dentro de la oración ingresada por el usuario. En este caso se están tomando en cuenta dos errores, el primero es el uso incorrecto del verbo "to be" y el segundo verifica si la conjugación del verbo en presente en las oraciones afirmativas es correcta.

Para el primer caso, cada pronombre tiene su forma del verbo "to be" particular, por ello en este caso lo que se hace es verificar que el sujeto tenga su forma adecuada. En el caso que sean sustantivos (no pronombres), se verifica si está en plural, singular, o existe más de un sujeto en la oración, los cuales están unidos por "and" y dependiendo del caso se verifica que se tenga la conjugación correcta.

"They *is* eating pasta."

En el caso de que la oración mostrada arriba, se tiene "is", después del pronombre "they", esto es un error gramatical, pues el verbo "to be" no se esta utilizando de forma adecuada, dicho error es identificado por el algoritmo y se envía un mensaje indicando un error del verbo to be.

Para el segundo error, se busca en la oración ingresada si alguno de sus respectivos tokens se clasificó como verbo de la tercera persona del singular. En el caso de que exista dicho token, se verifica quién es el sujeto de la oración, si es diferente de los sujetos de la primera persona del singular o de los sustantivos en su forma singular se determina que existe una incorrecta conjugación del verbo y por lo tanto existe un error gramatical, el cual se indica como "To be conjugation". Tal sería el caso de la oración siguiente:

"We *cooks* in the morning."

En este caso el verbo "cook"se esta conjugando para la tercera persona del singular (cooks) lo cual no es correcto pues se está hablando de nosotros "we", es decir de la primera

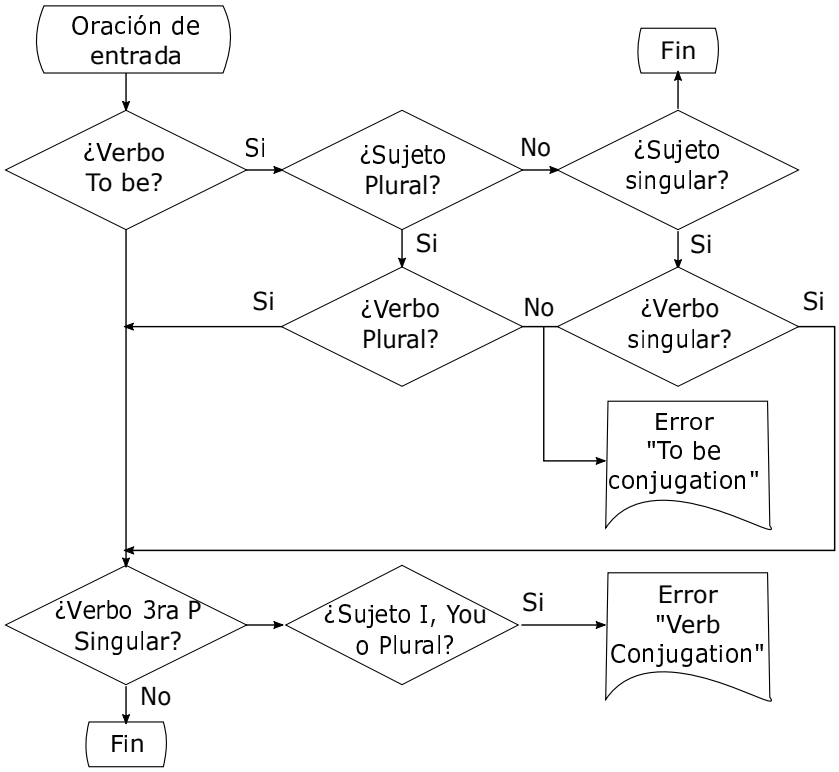


Figura 3.17: Diagrama de la clase GrammarMistake, encuentra errores en la oración.

persona del plural, por lo tanto esta oración no es correcta, tiene un error gramatical, el cual es identificado por el algoritmo y responderá con un mensaje "Verb conjugation"haciendo referencia a este error.

Para entender cómo funciona este proceso se presenta el diagrama en la Figura 3.17, el cual describe este proceso y además, se muestra el diagrama UML en la Figura 3.18.

3.4.6. Tiempos verbales (VerbTenses)

Esta clase permite definir los diferentes tiempos verbales que se utilizan en el inglés, con el fin de poder verificar que el o los verbos que están presentes en una frase verbal sean correctos.

El analizador lexicográfico junto con la clase de token y lematización son los encargados de etiquetar las palabras ingresadas por el usuario, pero la información que se tiene sobre un verbo es que este es verbo, que está en presente, pasado, en gerundio entre otros. Esta información no es suficiente, pues de la misma forma en que los humanos saben escribir oraciones más complejas que aquéllas que utilizan un solo verbo, la máquina debe ser capaz de hacer esto. Para ello, se definen los diferentes tiempos verbales existentes para que la máquina pueda ubicar el o los verbos en alguno de los tiempos verbales existentes y no se limite a tiempos verbales que sólo tienen un verbo en su estructura.

Por ejemplo, con esta clase se puede clasificar el conjunto de verbos "have been working"

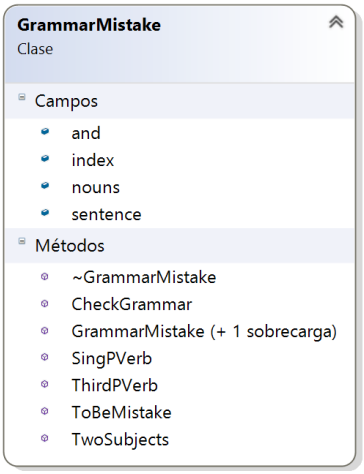


Figura 3.18: Diagrama UML de la clase “GrammarMistake”.

como presente perfecto continuo. Como se ve, se tienen tres verbos de tres posibles clasificaciones diferentes. Esta clase toma esos tres verbos y busca una coincidencia de ellos con los diferentes tiempos verbales, si la encuentra, indica que es una estructura correcta y se pueden considerar como el verbo de la frase verbal.

En esta clase también se permite clasificar las frases verbales negativas, y las abreviaciones posibles de dichas frases.

Esta clase posee los métodos y variables que se muestran en el diagrama UML de la Figura 3.19.

3.5. Casos

Con fin de mejorar los resultados arrojados por los algoritmos realizados en esta investigación se toman en cuenta varios casos particulares que se pueden presentar en las diferentes oraciones en inglés al momento de utilizar las reglas de la gramática libre de contexto o por algunas reglas gramaticales del inglés.

En el caso de las reglas de la gramática libre de contexto, se puede observar que al momento de llevar las mismas a la programación se va a obtener recursión, pues algunas funciones se van a llamar a sí mismas indirectamente.

En el caso de NP, cuando se verifica si una frase es NP, la misma puede invocar a la función PP y a su vez esta función de PP puede llamar a NP. Lo mismo sucede con PP, que llama a NP y NP llama a PP.

Una frase verbal o sustantiva puede contener más de una frase de preposición, por lo

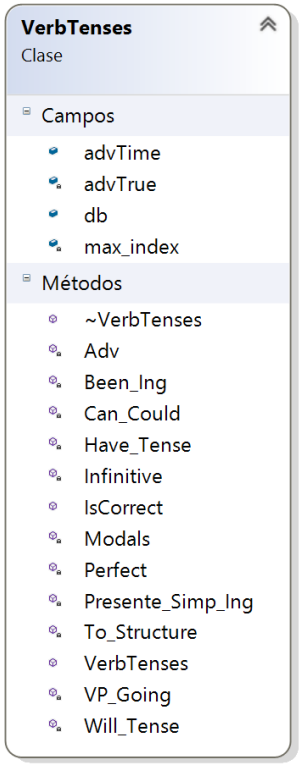


Figura 3.19: Diagrama UML de la clase "VerbTenses".

que la función PP tiene que ser capaz de manejar este caso al momento de presentarse. Por ejemplo si se observa esta oración:

"The gun was on the table near the window in the bedroom."

Se puede apreciar que existe recursión para la regla de la frase de preposición, que se repite en este caso tres veces. El algoritmo se hizo de forma que pueda aceptar este tipo de situaciones.

Según la reglas de CFG (Context Free Grammar) una frase sustantiva puede tener un determinador al inicio de la frase, pero gramaticalmente esto no siempre es correcto. Como se mencionó en la Subsección 2.4.2, los pronombres y los nombres propios no le preceden artículos, que en este caso se manejan dentro de los determinadores [25]. Por esta razón, no debe ser aceptado este tipo de estructura en las oraciones, sólo deben aparecer los determinadores antes de un sujeto que no pertenezca a ninguna de estas dos subcategorías.

Para cumplir con esto, en la función de frase sustantiva (NP), se agregó una regla. Cuando la palabra actual tiene coincidencia con algún determinador entra en este condicional y luego se observa si la palabra siguiente es un pronombre o un nombre propio, si lo es, se retorna falso pues la oración no está cumpliendo con la regla mencionada anteriormente y así se evita que se procese una oración que no es correcta.

Capítulo 4

Pruebas y Resultados

En esta tesis del procesamiento del lenguaje natural se tenía como objetivo general analizar oraciones en inglés utilizando procesamiento del lenguaje natural. Para lograr dicho objetivo se definieron objetivos más específicos que permitieron en conjunto alcanzar el objetivo buscado. Cada objetivo específico se conformó por ciertas tareas y pasos. Estos objetivos fueron la creación de un analizador lexicográfico, ubicación de un error gramatical y responder a las preguntas ¿Quién?, ¿Qué? y ¿Cuándo?.

A continuación se presentan y describen los resultados obtenidos en cada uno de estos objetivos que finalmente permiten el análisis de las oraciones en inglés.

4.1. Analizador Lexicográfico (Resultados)

El analizador lexicográfico es esencial en este trabajo de tesis, pues el mismo se puede considerar como el punto de partida para hacer otro tipo análisis a una oración. Por la metodología definida en esta tesis, sin la información que se obtiene en este objetivo no sería posible hacer ningún tipo de análisis pues las siguientes evaluaciones dependen de la información generada por el mismo.

Como se habló en Capítulo 3, el analizador lexicográfico en esencia toma cada palabra por el usuario y le asigna una o varias categorías léxicas, esta categoría en gran parte depende de la base de datos, pero se debe recordar, que también se tiene la clase Lemmatization, que se encarga de buscar formas modificadas de la palabra base, que no están en la base de datos, pero gracias a esta clase se pueden ubicar en una categoría.

Para evaluar y verificar este proceso se hizo uso de una interfaz gráfica desarrollada para esta tesis, con la herramienta Wintempla, que muestra la clasificación de cada palabra ingresada en la oración. En este proceso se necesita que las palabras sean clasificadas correctamente, sin importar si la entrada ingresada tiene sentido o por el contrario carece del mismo.

En la interfaz gráfica, se tiene un espacio para ingresar el texto, y otro para visualizar la

salida (clasificación). La salida mostrada o la clasificación que se iba a obtener era una de las siguientes:

- Adj : Adjetivos
- Adv: Adverbios
- Noun: Sustantivo
- Verb: Verbo
- Deter: Determinador
- Prepo: Preposición
- Int: Número Entero
- Double: Número racional
- Sepa: Separador*
- Time: Tiempo
- Unknown: Sin clasificación

En este caso, la clasificación de separador sólo hace alusión a la coma, sin embargo se podrían incluir fácilmente más caracteres que se consideren como separador en caso de que así se quisiera.

Se puede ver que se tienen menos clasificaciones que las definidas en Subsección 3.4.1, esto es porque en este caso se están agrupando varias subcategorías en la categoría general léxica a la que corresponde. Por ejemplo, sustantivos contables y no contables tanto singular como plural, pronombres y nombres propios se agrupan en sustantivos, lo mismo pasa con las demás subcategorías. Internamente se sigue manteniendo la información de las o la subcategoría exacta a la que pertenece la palabra.

En el Capítulo 3, se habló de que algunas palabras no se clasifican, pues no están en la base de datos, no son números, separadores, tiempo y tampoco son una forma modificada de alguna palabra base (lema). Para ello está la categoría desconocido, en donde se van a clasificar las palabras que no se logran clasificar. En lo posible se busca que las palabras no clasifiquen allí, pues si una palabra queda como desconocida no se puede analizar la oración.

Para cumplir con todo lo que se buscaba alcanzar en este objetivo en particular, se utilizó en conjunto las clases TokenDef, Lemmatization y Lexical A. En Sección 3.4 (clases), se habló en profundidad de la tarea y funcionamiento de cada una de ellas.

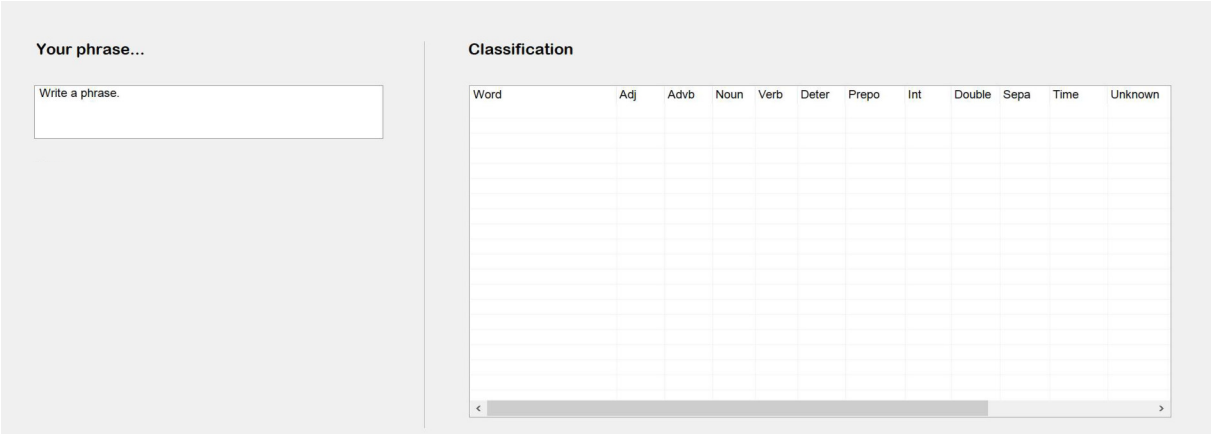


Figura 4.1: Interfaz gráfica utilizada para verificar el funcionamiento del analizador lexicográfico.

Puesto que una palabra puede pertenecer a varias categorías léxicas, se puede apreciar esto en la interfaz gráfica, para poder verificar este caso cuando suceda.

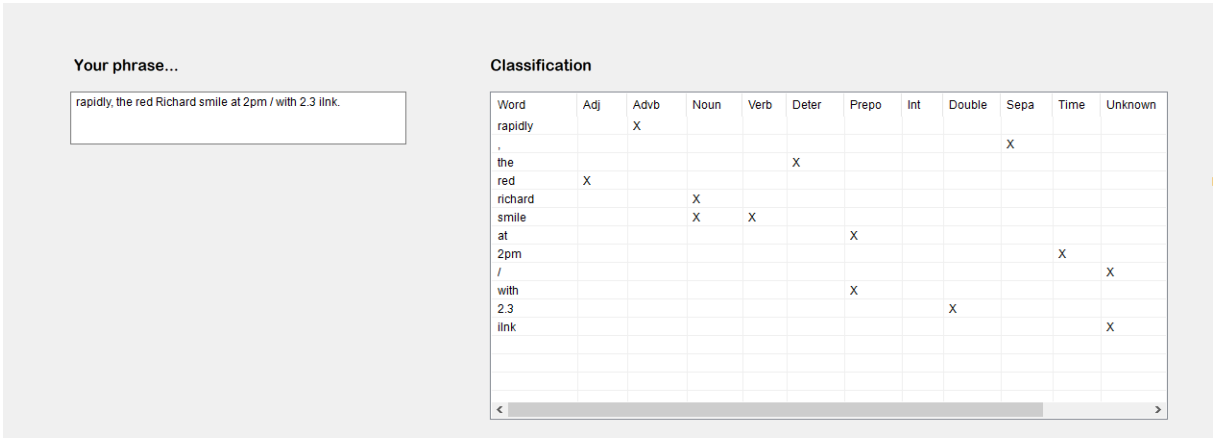
Se puede ver en la Figura 4.1 todas las clasificaciones mencionadas en la lista anterior. Se pueden ver también dos partes en la imagen, un recuadro de entrada y uno de salida. El de entrada es donde dice "Your Phrase" y el de salida "Classification". En el primer cuadro se escribe el texto y en el segundo cuadro se puede ver la clasificación de cada uno de los tokens que se encontraron en el texto ingresado.

Esta interfaz se utilizó todo el tiempo mientras se probaba el analizador lexicográfico, hasta obtener la clasificación correcta de todas las palabras para poder empezar con los otros análisis de las oraciones. En este caso el texto ingresado para hacer las pruebas no necesariamente tenía que ser oraciones, pues lo que se quería probar era la correcta clasificación de las palabras.

Se ingresaron varios grupos de palabras para verificar el correcto funcionamiento del algoritmo. Alguno de esos resultados se pueden ver en la Figura 4.2, Figura 4.3 y Figura 4.4.

En estas figuras se puede apreciar cómo se clasificó cada una de las palabras en una o más categorías. En la Figura 4.2 se tiene un texto sin coherencia, pero se puede apreciar cómo todos sus elementos se clasificaron en cada una de las categorías. En este caso sólo hay una categoría para la mayoría de las palabras, la excepción es "smile"; se puede ver una sola "x" en la línea correspondiente a cada palabra y dos a la palabra "smile".

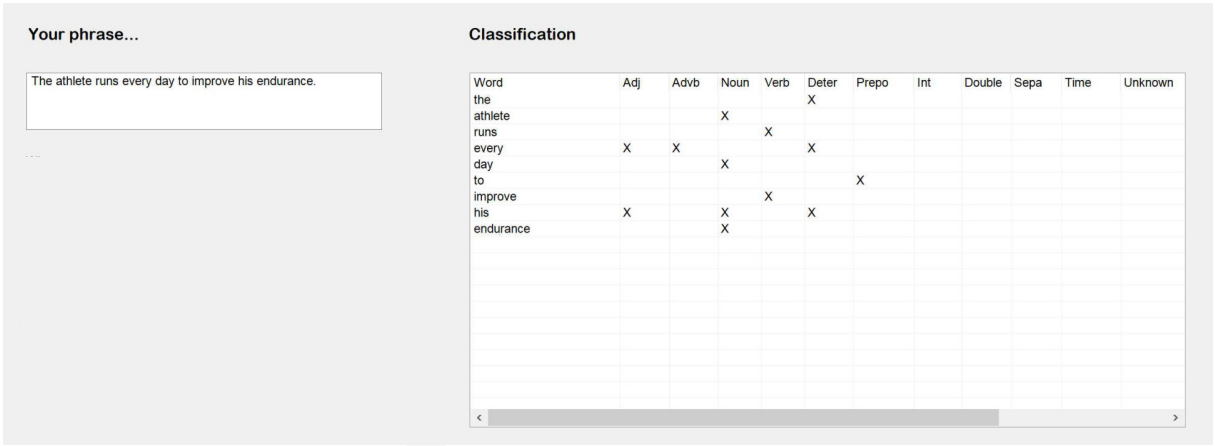
Como se ve en la Figura 4.2, se tienen dos tokens sin clasificación los cuales son, "/" e "ilnk". El primero es un símbolo que no se está trabajando, no se considera que en una oración debe estar dicho símbolo. La palabra ilnk no existe. El resto de las palabras se puede observar que tienen sus respectivas clasificaciones. El caso de "smile", a diferencia de las demás, se clasifica en dos categorías, verbo (verb) y sustantivo (noun). Todas estas clasifica-



Word	Adj	Advb	Noun	Verb	Deter	Prepo	Int	Double	Sepa	Time	Unknown
rapidly		X									
,									X		
the					X						
red	X										
richard			X								
smile			X	X							
at						X					
2pm										X	
/											X
with						X					
2.3								X			
ilnk											X

Figura 4.2: Analizador lexicográfico utilizando todas sus clasificaciones posibles en un texto de entrada.

ciones serán utilizadas posteriormente por otros algoritmos.



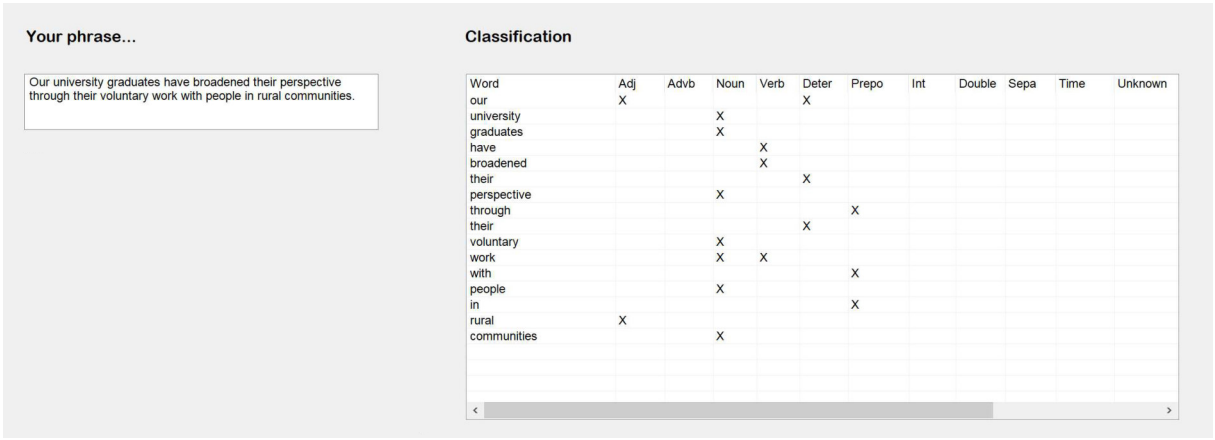
Word	Adj	Advb	Noun	Verb	Deter	Prepo	Int	Double	Sepa	Time	Unknown
the					X						
athlete			X								
runs				X							
every	X	X			X						
day			X								
to						X					
improve				X							
his	X		X		X						
endurance			X								

Figura 4.3: Etiquetado de todas las palabras de una oración.

En la Figura 4.3 se puede ver cómo se clasifican todas las palabras de la oración, se observan cómo algunas palabras tienen una clasificación, otras dos y otras hasta tres clasificaciones. Todas estas clasificaciones se pueden verificar con un diccionario, cualquier diccionario de inglés que indique el tipo de categoría a la que puede pertenecer la palabra consultada.

También se verifica que se puedan clasificar palabras que incluso no estén en la base de datos, esto con la ayuda de la clase de lematización. Este resultado se puede observa en la figura Figura 4.4. En esta figura se observa que se tiene la palabra communities, si se busca esta palabra en la base de datos, no se va conseguir, pero si existe la palabra “community”, debido a la clase de lematizacion, se puede determinar que estas palabras estan realacionadas, siendo “community” el singular de “communities”. Por esta razón se logra clasificar la palabra.

Finalmente, después de probar con 10 textos, similares a los mostrados en la Figura 4.2, Figura 4.3 y Figura 4.4. Se obtuvo un resultado satisfactorio con el algoritmo desarrollado. Es importante recordar que hay palabras que no se van a clasificar, por varias razones, estas son, porque no están en la base de datos, no son una palabra derivada de otra que sí esta en la base de datos o esta en una categoría que no se tomó en cuenta para este proyecto.



Word	Adj	Advb	Noun	Verb	Deter	Prepo	Int	Double	Sepa	Time	Unknown
our	X				X						
university			X								
graduates			X								
have				X							
broadened				X							
their					X						
perspective			X								
through						X					
their					X						
voluntary			X								
work			X	X							
with						X					
people			X								
in						X					
rural	X										
communities			X								

Figura 4.4: Etiquetado de otra oración con el analizador lexicográfico.

4.2. Análisis de oraciones

En este apartado se pretende mostrar los resultados obtenidos utilizando el algoritmo para extraer el qué, quién y cuándo de las oraciones analizadas.

Se manejó una interfaz gráfica, similar a la del apartado anterior, pero en este caso se tienen tres recuadros extras, en donde cada uno de ellos muestra el Qué, Quién y Cuándo de la oración ingresada. Se debe tener en cuenta que el Cuándo de la oración no es un elemento que siempre va a estar presente, por lo tanto en dichos casos donde no exista, la respuesta será “none”.

En cuanto al Quién y el Qué, si deben estar presentes en todas las oraciones, pues en este trabajo se están analizando oraciones declarativas positivas y negativas. Ambas oraciones tienen un sujeto, y un verbo explícitos.

Para poder evaluar este objetivo se buscaron oraciones en inglés de diferentes niveles, con diferentes tiempos verbales y con un vocabulario diverso.

Con el fin de mostrar un poco la interfaz y la salida que se obtiene se muestran algunos ejemplos. Para empezar se muestra una oración en la Figura 4.5 de tipo declarativa de donde se puede extraer la información necesaria para contestar las tres preguntas. Se tiene un sujeto

que es Tom, en este caso es el “Who”, un tiempo verbal, presente continuo “is sleeping”, respondiendo al “What”, y finalmente “now” responde a “When”. En este ejemplo se puede observar que la respuesta al Qué, esta compuesta por dos verbos, esto se logra con la ayuda de la clase “verbTenses”.

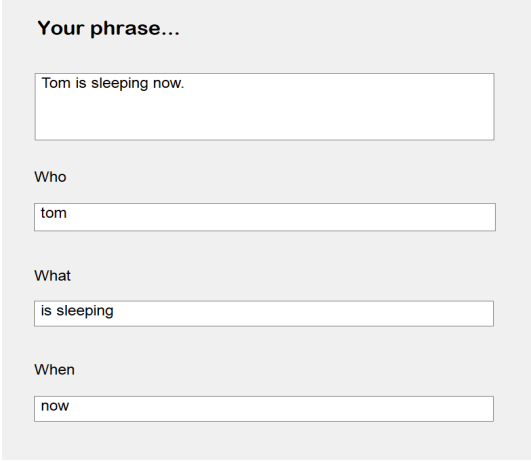


Figura 4.5: Oración en presente continuo.

En la figura la Figura 4.6, se puede ver otro ejemplo, donde también se responde a las tres preguntas. Se puede observar que el tiempo verbal de esta oración es en pasado simple y de igual forma se analiza la oración sin problemas.

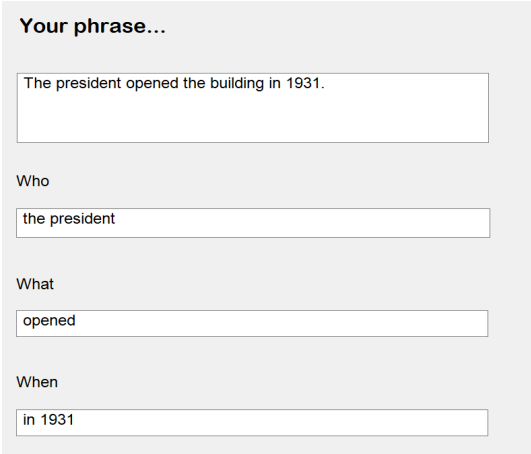


Figura 4.6: Oración en pasado simple.

En Figura 4.7, se puede ver cómo se halla el Qué de otra oración, cuyo tiempo verbal está compuesto por una serie de verbos, en este caso se usa el tiempo verbal futuro perfecto continuo (future perfect continuous). En este ejemplo, se puede ver que el When tiene como respuesta “None”, esto porque la oración no tiene respuesta al Cuándo.

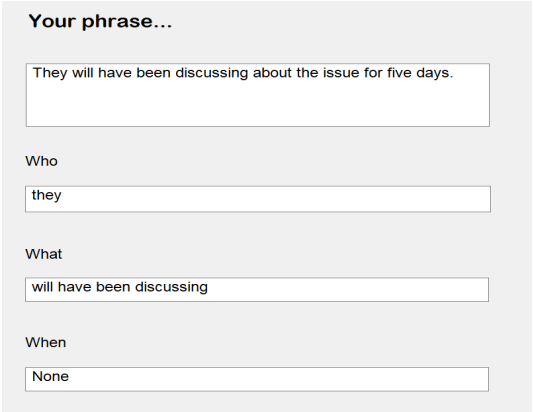


Figura 4.7: Oración en “future present continuous”.

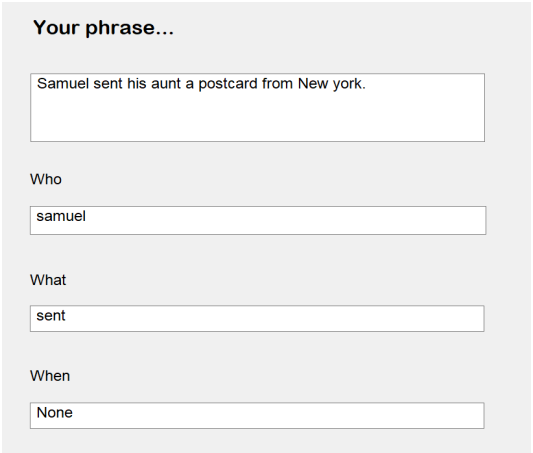


Figura 4.8: Oración con objeto directo e indirecto.

El algoritmo también permite evaluar oraciones con dos objetos, así se pueden evaluar oraciones con objeto directo e indirecto como por ejemplo:

“Samuel sent his aunt a postcard from New york.”

En esta oración “his aunt” es el objeto indirecto y “a postcard” es el objeto directo, la evaluación de esta oración se hace sin problemas con el algoritmo, esto se puede observar en la figura Figura 4.8.

El “and” también aparece en algunas ocasiones uniendo frases sustantivas, esto se puede ver en el ejemplo de la Figura 4.9, donde el Quién de la oración, es “Luis and Maria”. Con el algoritmo, se pueden evaluar oraciones compuestas si la conjunción que las une es “and”.

De esta forma se continúan evaluando las diferentes oraciones, verificando que se obtenga una respuesta correcta a las preguntas mencionadas.

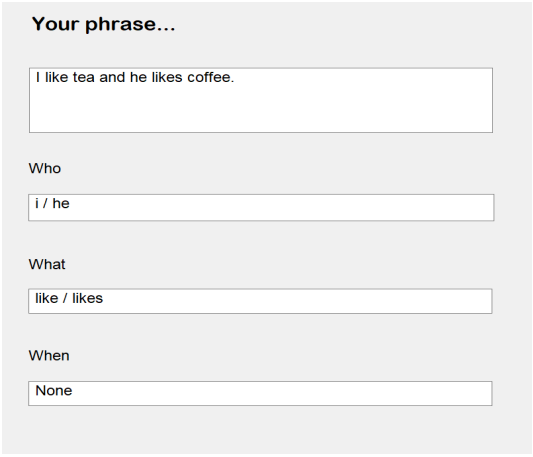


Figura 4.9: Oración compuesta (compound sentence).

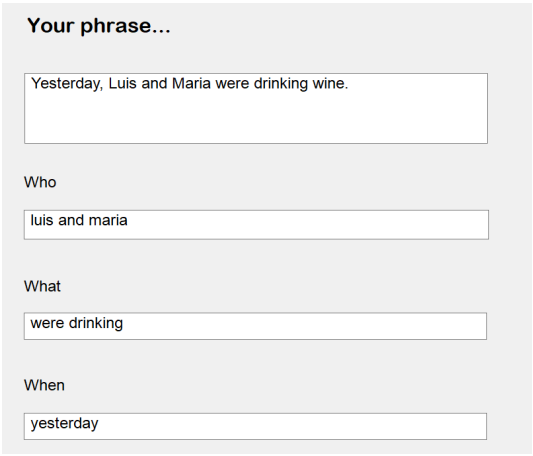


Figura 4.10: Oración con dos sujetos unidos por “and”.

Se probaron 150 oraciones declarativas tanto positivas como negativas, para observar el análisis de las mismas en el algoritmo desarrollado. Estas oraciones fueron extraídas de [30] y recursos de internet [26], [31]. El resultado se puede apreciar en la Tabla 4.1. Como se ha mencionado anteriormente, las oraciones analizadas tienen sujeto y verbo por ser oraciones de tipo declarativas, pero no necesariamente tienen alguna frase o palabra que responda al cuándo. Por lo tanto el resultado mostrado toma como correctas oraciones en donde el cuándo tienen un “None” como respuesta, en caso de que no exista una palabra o frase que pueda responder a la pregunta.

Para observar los resultados de forma mas fácil, se toman los porcentajes. Se puede ver que el porcentaje más alto, la pregunta que tuvo más respuestas es el Quién, seguido de el Qué y finalmente el Cuándo. Esto se puede observar el la Tabla 4.2.

	Todas las respuestas	¿Quién?	¿Qué?	¿Cuándo?
Correcta	109	117	115	109
Incorrecta	41	33	35	41

Cuadro 4.1: Resultado de las oraciones analizadas

	Todas las Respuestas	¿Quién?	¿Qué?	¿Cuándo?
Correcta	72.66 %	78 %	76.66 %	72.66 %
Incorrecta	27.33 %	22 %	23.33 %	27.33 %

Cuadro 4.2: Resultados de las oraciones analizadas en porcentaje.

4.3. Análisis del error Gramatical

En esta sección se muestran algunos ejemplos realizados para mostrar la capacidad del algoritmo de hallar errores gramaticales en las oraciones ingresadas así como los resultados finales que se obtuvieron.

Como se mencionó en la Subsección 3.4.5, los errores que se analizan en este trabajo son errores en la conjugación del verbo "to be" y en la conjugación de los verbos en presente.

Para hacer el análisis se tomaron oraciones que utilizan el verbo "to be" y otras que utilizan los verbos en presente simple de forma correcta, y estas mismas oraciones se les modificó el verbo de manera que su conjugación pasara a ser incorrecta.

Con estas oraciones y utilizando una interfaz gráfica que indica si existe error y de qué tipo se realizó el análisis respectivo. En el caso de ser un error referente al verbo "to be", se muestra "to be conjugation", y en el caso de ser un error con respecto a la conjugación de cualquier otro verbo en su forma de presente simple se muestra "Verb conjugation". Ambos mensajes se muestran en el recuadro que se llama "Mistake" en la interfaz gráfica.

En la Figura 4.12, se puede observar lo mencionado en el párrafo anterior. En el primer recuadro se puede ver la oración y un mensaje que indica que existe un error en la conjugación del verbo"to be". En el segundo recuadro aparece la misma oración pero esta vez de forma correcta, y se puede observar que no aparece ningún mensaje de error pues en esta segunda oración el verbo "to be" coincide con el sujeto.

Para el caso en que se presente un error del otro tipo mencionado, se obtiene una salida como la que se muestra en la Figura 4.12. En la figura se puede ver la oración en el primer recuadro y en el recuadro de Mistake se indica "verb conjugation".

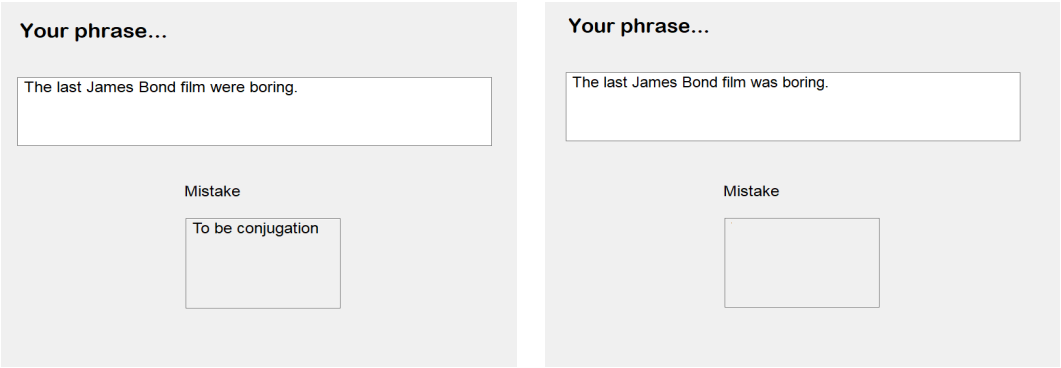


Figura 4.11: Error en el verbo To be.

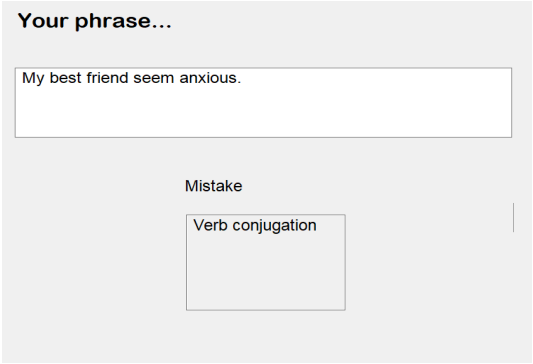


Figura 4.12: Error en un verbo de tipo presente simple.

Para evaluar la respuesta del algoritmo al momento de hallar errores gramaticales de conjugación del verbo "to be" y del verbo en presente simple, se ingresaron 24 oraciones que utilizaban el "to be" ya sea en presente o pasado y 24 oraciones en presente simple. Para ambos casos, la mitad de las oraciones estaban correctas y la otra mitad eran las mismas oraciones pero con el verbo de forma incorrecta.

	To be	Presente Simple
Correctas	95.83 %	79.16 %
Incorrectas	4.16 %	20.83 %

Cuadro 4.3: Resultados obtenidos al hallar errores gramaticales en las oraciones.

Se consideraron correctas las oraciones en donde no se indicó error gramatical y no existía ningún error gramatical de los tipos que se tratan en este trabajo y también las oraciones en donde sí se presenta un error gramatical y el mensaje de error indico dicho error. Con estas pruebas se obtuvieron los resultados que se muestran en la Tabla 4.3. En dicha tabla se muestran los porcentajes con respecto a los dos grupos de 24 oraciones probadas. En el caso del verbo "To be" el 95.83 % de las oraciones se evaluó de forma correcta, es decir cuando había error gramatical se indicó la presencia del mismo y cuando no existía no se indicó ningún tipo de error.

Capítulo 5

Conclusiones y Trabajos Futuros

El PLN es un área que está tomando auge en los últimos años y además se están identificando muchas aplicaciones para el mismo. Por esto, se debe continuar la investigación en esta área pues es un tema que aún tiene mucho por explorar y desarrollar.

En este capítulo se presentan las conclusiones de este trabajo sobre el procesamiento del lenguaje natural, así como los posibles trabajos futuros para extender el alcance del presente proyecto.

5.1. Conclusiones

El PLN es un área bastante extensa que tiene varias ramas las cuales pueden ser trabajadas. En esta tesis se utilizó PLN en el análisis léxico, de gramática y sintáctico, abarcando así varios aspectos de esta compleja área.

Se logró identificar los diferentes tokens y las diferentes clasificaciones de las palabras que conforman la oración. De igual forma se logró manejar los verbos que constan de dos partes, conocidos como "phrasal verbs" y aceptarlos como un token más.

Con este algoritmo se puede extraer el Qué, Quién y en varios casos, si existe, también el Cuándo de la oración ingresada en caso de que dicha oración tenga un adverbio que identifique el tiempo.

Se pueden analizar oraciones que contengan dos objetos del tipo frase sustantiva (noun phrase), aunque esto no está en las reglas básicas del CFG, se realizó el algoritmo de tal forma que lo permita y analice este tipo de oraciones, de esta forma se extiende el alcance.

Se pueden analizar oraciones simples y compuestas. Las oraciones compuestas que se pueden analizar son aquéllas que están unidas por "and".

El algoritmo permite analizar oraciones enunciativas tanto positivas como negativas. Dentro de estas oraciones se pueden manejar las oraciones en pasado, presente,futuro y sus deri-

vaciones como continuo y perfecto en los respectivos casos. Así como también oraciones que tienen verbos auxiliares y de modo.

Mediante este algoritmo es posible identificar cuando se presenta un error de conjugación con el verbo "to be" además se puede determinar si el resto de los verbos (diferentes al verbo to be) en presente, no están conjugados de manera correcta de acuerdo al sujeto que le precede. En los resultados obtenidos se ve que se la probabilidad de hallar el error en el verbo "to be" es mayor a la de los otros verbos en presente simple.

Es muy importante tener un gran conocimiento de la estructura y el vocabulario del lenguaje a analizar con anterioridad, para poder aplicar métodos como los realizados en esta tesis, de lo contrario se van a tener muchos problemas a lo largo del desarrollo de los algoritmos y reglas definidas en ellos.

5.2. Trabajos Futuros

Debido a la complejidad del lenguaje se pueden agregar muchas cosas más a estos algoritmos para abarcar otros tipos de oraciones y estructuras.

Por ejemplo, se pueden agregar los adjetivos comparativos y superlativos ya que se presentan frecuentemente en las oraciones y así se permitirá el análisis de muchas más oraciones. Así como este tipo de palabras, se pueden agregar muchas otras, como por ejemplo las conjunciones (además de and) y subclasificación de los tipos sintácticos de las palabras.

También se pueden agregar otras frases, como por ejemplo las frases adverbiales, incluyendo este tipo de frase se puede responder al ¿Cuando? de muchas mas oraciones. Este tipo de frase agrega una gran complejidad al algoritmo por lo que es un punto que se tiene que revisar con mucho detenimiento.

Se puede adecuar el algoritmo para permitir otro tipo de oraciones compuestas (compound sentences), si además del "and" se incluyen las conjunciones (conjuction) restantes que permiten formar otras oraciones de este tipo.

Partiendo de la metodología utilizada en la clase Lemmatization, se podrían generar los verbos en pasado y por ende en participo pasado de los verbos regulares. Estos en su mayoría, terminan de una forma común, agregando esto al código no habría necesidad de tener una base de datos para este tipo de verbos.

Para hacer análisis más exhaustivos del lenguaje o dar más capacidad a la máquina de procesar el lenguaje como los humanos, sería bueno combinar diferentes técnicas de PLN. Esto debido a la complejidad que presenta por sí solo el lenguaje, combinando diferentes técnicas se puede manejar diferentes áreas del lenguaje y así no depender netamente de reglas.

En trabajos posteriores se puede considerar manejar bigramas no sólo para los phrasal verbs, sino para más tipos de palabras, como por ejemplo los adverbios de tiempo. Además se pueden manejar n-gramas, con n mayor a dos, con ello procesar de manera conjunta palabras que aparezcan frecuentemente juntas como si fuera una sola palabra.

Para un trabajo futuro, se puede considerar utilizar word2vec, para que el algoritmo conozca más palabras de las que están en la base de datos de los archivos .txt. Así se podrá ejecutar el algoritmo completamente incluso si alguna de las palabras en la oración no está en la base de datos .txt.

Word2vec se entrena y después de dicho entrenamiento puede determinar la similitud entre palabras, así, si se consigue una palabra que no se tenga en la base de datos dentro de la oración que se está analizando se podría buscar si es muy similar a otra palabra que si está en la base de datos, en caso de que se tenga una palabra similar, se analiza la oración con dicha palabra sinónimo de la desconocida.

En el lenguaje natural existe algo llamado "World knowlegde", conocimiento del mundo que rodea al ser humano. Esto es lo que permite a los seres humanos realizar oraciones como la siguiente:

"The boy is running like a gazelle."

En esta oración si se quiere contestar cómo corre el niño, los humanos por el conocimiento del mundo que tiene, saben que el niño corre rápido, pues se conoce que las gacelas corren rápido, pero debido a que la máquina no tiene este conocimiento probablemente no sería capaz de saber y no conectaría el "like a gazelle" con el niño o con el verbo correr.

Los algoritmos realizados en el área de procesamiento del lenguaje natural reciben este tipo de oración y no tienen forma de determinar exactamente su significado pues no tiene conocimiento del mundo que lo rodea (al menos en las investigaciones realizadas no se consiguió un método que sí lo pueda hacer). En este caso simplemente se analizará la oración como cualquier otra. Este tipo de casos son temas que se deben seguir investigando y tratando, para buscar una forma de solucionarlo, y darle la posibilidad a la computadora de tener ese "conocimiento del mundo"que tenemos los seres humanos y de esta forma ampliar la capacidad de las máquinas para entender el lenguaje natural.

Bibliografía

[1] "Asociación mexicana para el procesamiento del lenguaje natural. [En línea]. Disponible: http://www.cicling.org/ampln/."

[2] N. Erbs, P. B. Santos, I. Gurevych, and T. Zesch, "DKPro Keyphrases: Flexible and Reusable Keyphrase Extraction Experiments," *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 31–36, 2014.

[3] R. Dale, H. Moisl, and H. Somers, *Handbook of Natural Language Processing*. CRC Press, 2000.

[4] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," *Proceedings - International Conference on Software Engineering*, pp. 499–508, 2007.

[5] M. Osborne, S. Moran, R. McCreadie, A. V. Lunen, M. Sykora, E. Cano, N. Ireson, C. Macdonald, I. Ounis, Y. He, T. Jackson, F. Ciravegna, and A. O'Brien, "Real-Time Detection , Tracking , and Monitoring of Automatically Discovered Events in Social Media," *The 52nd Annual Meeting of ACL*, pp. 37–42, 2014.

[6] C. Lipizzi, L. Iandoli, and J. E. Ramirez Marquez, "Extracting and evaluating conversational patterns in social media: A socio-semantic analysis of customers'reactions to the launch of new products using Twitter streams," *International Journal of Information Management*, vol. 35, no. 4, pp. 490–503, 2015.

[7] E. D. Liddy, "Natural Language Processing," in *Encyclopedia of Library and Information Science*, vol. 37, pp. 51–89, Marcek Decker, Inc., 2nd ed., 2001.

[8] G. G. Chowdhury, *Natural language processing*, vol. 37. University of Strathclyde, 2005.

[9] J. Aravind K, "Natural language processing," *Commun. ACM*, vol. 39, no. 1, pp. 60–62, 1996.

[10] C. D. Manning and H. Schutze, *Foundations of statistical natural language processing*, vol. 31. The MIT Press Cambridge, 2002.

[11] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, vol. 21. Prentice Hall, 2009.

[12] W. J. Gillis, S. Khan, C. J. Schmidt, and K. Vijay-Shanker, "Signal Transduction: NLP Based Information Extraction Using a Multi-agent System," pp. 1–11.

[13] C. D. Manning, J. Bauer, J. Finkel, S. J. Bethard, M. Surdeanu, and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, 2014.

[14] T. Nasukawa and J. Yi, "Sentiment analysis: Capturing favorability using natural language processing," *Proceedings of the 2nd international conference on Knowledge capture*, pp. 70–77, 2003.

[15] E. Riloff, A. Qadir, P. Surve, L. D. Silva, N. Gilbert, and R. Huang, "Sarcasm as Contrast between a Positive Sentiment and Negative Situation," *Emnlp*, no. Emnlp, pp. 704–714, 2013.

[16] R. Huang and E. Riloff, "Classifying Message Board Posts with an Extracted Lexicon of Patient Attributes," *Emnlp*, no. October, pp. 1557–1562, 2013.

[17] Y. Sun, J. Leigh, A. Johnson, and S. Lee, "Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations," pp. 184–195, 2010.

[18] M. Mohri, "On Some Applications of Finite-State Automata Theory to Natural Language Processing Mehryar Mohri," vol. 1, no. 1, pp. 61–80, 1996.

[19] T. M. Jones, "AI Application Programming (Programming Series)," 2003.

[20] E. Roche and Y. Schabes, *Finite-state Language Processing*. MIT Press, 1997.

[21] M. Newson, *Basic English Syntax*. Bölcsész Konzorcium, 2006.

[22] J.-b. Kim and P. Sells, *English Syntax : An Introduction Of Language*. CSLI publications, 2008.

[23] "Oxford Living Dictionaries. [En línea]. Disponible: http://en.oxforddictionaries.com/."

[24] J. Eastwood, *Oxford Guide To English Grammar*. Oxford University Press, 1994.

[25] G. Yule, *The Study of Language*. Cambridge University Press, fourth edi ed., 2010.

[26] "British Council: Learn English. [En línea]. Disponible: https://learnenglish.britishcouncil.org."

[27] University of Ottawa, "The Writing Centre," *[Online]. Disponible: http://arts.uottawa.ca/writingcentre/en*.

[28] J. Weidl, "The Standard Template Library Tutorial," *Information Systems Institute, Distributed Systems Department Technical University Vienna*, 1996.

[29] R. Kibble, "Introduction to natural language processing," *University of London*, 2013.

[30] R. Murphy, *English Grammar in Use*. Cambridge University Press, fourth edi ed.

[31] "EF: [En línea]. Disponible https://www.ef.com/english-resources/english-grammar/."