UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Adaptación de una herramienta de procesamiento de lenguaje natural para el etiquetado de sentimientos y el análisis de lenguaje en español

TRABAJO DE TITULACIÓN

AUTOR: Correa Cordero, Mario Francisco

DIRECTOR: Valdiviezo Díaz, Prisila Marisela, Ing.

LOJA - ECUADOR

2015

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Ingeniera.

Prisila Marisela Valdiviezo Díaz.

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de fin de titulación: Adaptación de una herramienta de Procesamiento de Lenguaje Natural para el etiquetado de sentimientos y el análisis de lenguaje en español, realizado por Mario Francisco Correa Cordero ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, noviembre de 2015

f).....

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Yo Correa Cordero Mario Francisco, declaro ser autor del presente trabajo de titulación: Adaptación de una herramienta de Procesamiento de Lenguaje Natural para el Etiquetado de Sentimientos y el Análisis de Lenguaje en Español, de la Titulación Sistemas Informáticos y de Computación, siendo Prisila Marisela Valdiviezo Díaz directora del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja, que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f).....

Autor: Mario Francisco Correa Cordero

Cédula: 1104064140

DEDICATORIA

Dedico este trabajo a mis padres Paquita Cordero y Mario Correa, pilares fundamentales en mi vida. Siendo un gran ejemplo a seguir y no solo para mi sino también para mis hermanos.

A mis hermanos, por haberme apoyado en cada etapa de mi vida.

A mis amigos y compañeros, aunque habiendo 7 millones de personas en el planeta y me ha tocado con ellos, además de haberme hecho entender la razón del porque Batman trabaja solo.

A mis profesores por el apoyo que me han brindado a lo largo de la carrera.

“Me aburro fácilmente y, sobre todo, para mí escribir no es trabajar.” — Neil Gaiman

AGRADECIMIENTO

Agradezco primeramente a mis padres y hermanos por el apoyo y comprensión incondicional.

A mis tutores Mgtr. Prisila Valdiviezo, Ing. Guido Riofrío y Mgtr. Rodrigo Barba por todo el conocimiento y apoyo que me brindaron.

A mis amigos y familiares, gracias por estar conmigo, por su fuerza, confianza y cariño.

A todas las personas que con su apoyo y enseñanza hicieron que este proyecto sea posible.

A Yesenia Ortiz y Diana Morocho, por brindarme su apoyo, y colaboración en todo momento, por el tiempo compartido y por la amistad que siempre me brindaron.

A la UTPL, por ser mi casa durante todo este tiempo y darme todas las facilidades para crecer y seguir adelante.

A Charbel y Freddy también les agradezco.

A todos aquellos familiares y amigos que no recordé al momento de escribir esto. Pero ellos saben que fueron importantes para el desarrollo de este proyecto. Simplemente gracias.

"Las palabras significan lo que nosotros queramos." — Neil Gaiman

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El objetivo de este trabajo de titulación fue realizar la adaptación de una herramienta de Procesamiento de Lenguaje Natural y el Etiquetado de Sentimientos en Español, basados en el análisis sistemático y lingüístico de texto.

En la actualidad existen varios programas que ayudan a realizar un procesamiento de lenguaje natural (PLN), sin embargo en este trabajo se utiliza OpenNLP debido a las ventajas que presenta como: Consume una menor cantidad de recursos y realiza el procesamiento en menor tiempo. OpenNLP una colección de proyectos distribuidos bajo licencia de código abierto, desarrollado en Java, que ofrece las siguientes herramientas: Tokenizador, detección de oraciones, reconocedor de la entidad y etiquetado gramatical.

Para la obtención de los datos, primero se separó el texto en oraciones. Cada frase fue dividida en tokens para asignarle una etiqueta gramatical. Ésta etiqueta gramatical fue la propuesta por el grupo EAGLES para los idiomas europeos, que incluye el idioma español. Además, al token se le asignó un etiquetado de emociones como: aburrimiento, angustia, ansiedad o preocupación, confusión, frustración y simpatía.

Finalmente, muchas de las herramientas que están disponibles en español son limitadas, por lo tanto se creyó conveniente implementar un instrumento con mayores funcionalidades para el beneficio de nuevas investigaciones.

PALABRAS CLAVE: PLN, token, OpenNLP, emociones, etiquetas, tokenizador, EAGLES.

ABSTRACT

The objective of the following certification work was to realize the adaptation of a Natural Process Language tool and the tagging of feelings in Spanish, based on the systematic and linguistic analysis from the text.

In the actuality it exists various programs which helps to realize a Natural Process Language (NPL), although in this work OpenNLP was used due to all the advantages that presents like: to consume a lower quantity of resources and preform the process in the shortest time. OpenNLP a project collection distributed under the license of open code, developed in JAVA, which offers the following tools: Tokenize, sentence detection, entity recognizer and grammatical tagging.

In order to obtain data, first we separated the text in sentences. Each phrase was divided in tokens in order to assign a grammatical tag. This grammatical tag was proposed by the group of EACGLES for the European languages, which also includes Spanish.

Also, a tag of emotions was assigned to the token such as: boredom, anguish, anxiety, preoccupation, concern, confusion, frustration and sympathy.

Finally, a lot of the tools that are available in Spanish are quite limited, therefore it was found convenient to implement a toll or and instrument with higher features for the benefit of new investigations.

KEYWORDS: PLN, Token, emotions, OpenNLP, tokenizer, tags, EAGLES.

INTRODUCCIÓN

El trabajo se encuentra enfocado en la adaptación de una herramienta de Procesamiento de Lenguaje Natural que permita analizar textos en español a través de árboles sintácticos, identificando la categoría de cada palabra, por ejemplo sustantivo, verbo, adjetivo, etc.

El Procesamiento de Lenguaje Natural, pretende identificar la información enviada a una máquina y que ésta logre entenderla y procesarla a la misma. El lenguaje que poseen los seres humanos está ligado al conocimiento lingüístico y contenido contextual que posee la persona.

A continuación se detallan los objetivos que se han considerado para el desarrollo de este proyecto:

- Realizar la adaptación de una herramienta de Procesamiento de Lenguaje Natural en el idioma español.
- Realizar el etiquetado de las palabras y la construcción de árboles de análisis sintáctico para el análisis de textos en español.
- Desarrollar e investigar herramientas de modelado y análisis, para el manejo de datos e información.
- Normalizar el texto utilizado para el análisis del lenguaje.
- Realizar el etiquetado de sentimientos de las palabras, y la construcción de corpus para su posterior análisis.

Este trabajo está estructurado de la siguiente manera:

Capítulo I. Estado del arte.- Descripción de los temas que se van a ocupar para el desarrollo de este trabajo.

Capítulo II. Análisis de las herramientas.- Se da una visión de la forma en la que algunas herramientas de PLN trabajan, para escoger una de estas y desarrollar este trabajo.

Capítulo III. Construcción de la herramienta.- se selecciona una de las herramientas analizadas en el capítulo II, se procede a la construcción de una herramienta para realizar el análisis del texto en español y realizar el etiquetado de sentimientos.

Capítulo IV. Resultados.- Considerando un banco de mensajes de la red social del entorno virtual de aprendizaje, se presenta los resultados obtenidos con la aplicación desarrollada.

CAPÍTULO I

ESTADO DEL ARTE

1.1. Introducción

Este capítulo comprende la parte teórica que será base para el desarrollo de este trabajo. Iniciando con el estado del arte que abarca tema como el Procesamiento de Lenguaje Natural algunas definiciones y ejemplos, que son necesarios para obtener un modelo de análisis sintáctico. Además de incluir contenidos que son relevantes para el desarrollo de este proyecto.

Se pretende obtener una base sólida que fundamente el desarrollo de un prototipo para realizar un análisis del lenguaje y un etiquetado de sentimientos.

1.2. Recuperación de información

La recuperación de información es un término que suele definirse en un sentido bastante amplio (van Rijsbergen, 1979), porque muchas veces se lo presenta como un concepto equivalente a la recuperación de datos, desde la perspectiva de una base de datos. Así, al mirar la contraportada de un juego y poder escribir el código de activación es una forma de recuperación de información. Sin embargo, se la puede definir de la siguiente manera:

(Manning, Raghavan, & Schütze, 2008) La recuperación de información se está transformando en una forma dominante para acceder a la información, superando a las búsquedas tradicionales en bases de datos. Podemos citar las siguientes definiciones:

La recuperación de la información consiste en encontrar el material en documentos de naturaleza no estructurada generalmente de texto, que satisfacen una necesidad de información desde dentro de grandes colecciones, generalmente almacenada en las computadoras (Manning et al., 2008).

La Recuperación de Información o Information Retrieval es la representación, almacenamiento, organización y acceso a ítems de información (Baeza Yates & Ribeiro Neto, 1999).

La recuperación de información era una actividad que pocas personas realizaban como: Asistentes legales, bibliotecarios, oficinistas o profesiones similares. En la actualidad miles de personas pueden realizar la tarea de recuperar información en cualquier momento, cuando utilizan un buscador web, al buscar algún correo electrónico importante en el buzón.

El problema fundamental que tiene la recuperación de información, es determinar cuáles son los documentos relevantes para el usuario por medio de una consulta.

(Lancaster, 1968) un sistema de recuperación de información no informa (modifica el conocimiento) al usuario sobre el tema de su investigación; informando sobre su existencia (o inexistencia) de los documentos solicitados.

1.2.1. Modelos de recuperacion de informacion

1.2.1.1. Modelo booleano

Este modelo se establece mediante las consultas clásicas de informacion que ofrecen los operadores lógicos como el AND, OR y NOT. El resultado del operador logico con la consulta presenta todos los documentos que satisfacen y conforman el conjunto de los documentos relevantes, particionando los documentos en conjuntos que cumplen y no cumplen la condición, parecido a lo que sucede con una base de datos tradicional (Vilares Ferro, 2005). Una representacion gráfica de este modelo se encuentra en la figura 1-1.

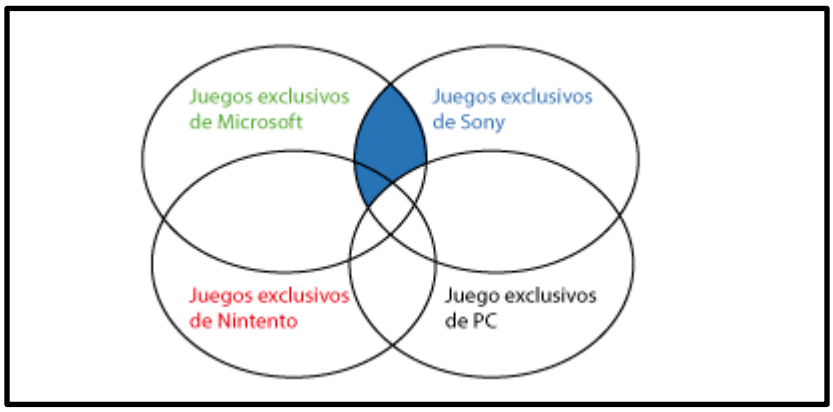

Figura 1-1. Modelo Booleano: Microsoft AND Sony AND Nintento AND NOT PC
Elaboración: Autor de la tesis

1.2.1.2. Modelo vectorial

Se fundamenta en la similitud que tienen los documentos por medio de la construcción de un vector de palabras claves, que posee cada documento de forma enlazada. Permite asignar pesos a las claves y obtener resultados con cierto grado de igualdad entre ellos, y su recuperacion es mejor que la del modelo boolenao; por esta razón es el modelo más usado en la recuperación de información (Vilares Ferro, 2005). Se observa una representación gráfica de este modelo en la figura 1-2.

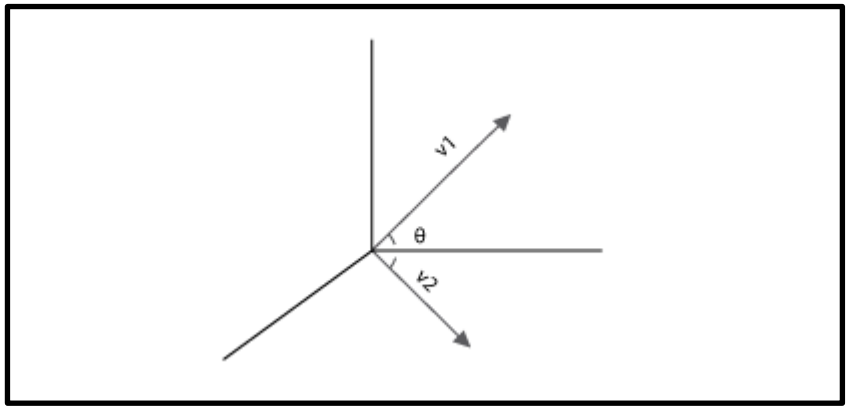

Figura 1-2. Modelo Vectorial
Fuente: (Vilares Ferro, 2005)
Adaptado: El autor de la tesis

1.2.1.3. Modelo probabilístico

El modelo probabilistico pretende solventar el problema que tiene la recuperacion de información, calculando la probabilidad de que un documento sea notable para una consulta (Vilares Ferro, 2005).

1.2.2. Recuperación de información vs Recuperación de datos

(van Rijsbergen, 1979) indica las diferencias entre Recuperación de Información y Recuperación de datos (Data Retrieval) las cuales se demuestran en la tabla 1-1.

Tabla 1-1. Recuperación de Información vs Recuperación de datos

Propiedad	**Recuperación de Datos**	**Recuperación de Información**
Coincidencia	Coincidencia exacta	Coincidencia parcial
Inferencia	Deducción	Inducción
Modelo	Determinista	Probabilístico
Clasificación	Monotética	Politética
Lenguaje de consulta	Artificial	Natural
Especificación de la consulta	Completo	Incompleto
Ítems solicitados	Coincidencia	Relevante
Respuesta de error	Sensibles	Insensibles

Fuente: (van Rijsbergen, 1979)

La tabla 1-1, demuestra que la recuperación de datos busca una coincidencia exacta, es decir ve si un elemento se encuentra en el archivo. Estas búsquedas suelen ser de interés para la

recuperación de información, que trata de hacer corresponder parcialmente los elementos con la consulta para seleccionar los mejores.

La recuperación de información usa una inferencia de inducción común, dejando que las relaciones se especifiquen solamente con el grado de certeza o incertidumbre, por lo tanto su confianza es variable. En cambio, la recuperación de datos ocupa una inferencia de deducción simple, por lo que podemos decir que AB y BC es igual a AC. Con esta distinción se describe la recuperación de datos como una recuperación determinista y la recuperación de información como probabilística.

Además, en la tabla 1-1 se aprecia otra diferencia en términos de clasificación. En la recuperación de datos interesa la clasificación monotética, donde las clases están definidas por objetos que poseen atributos suficientes para pertenecer a una clase, pero esta clasificación no es útil para la recuperación de información. En la clasificación politética cada individuo de una clase tendrá una porción de atributos que tienen todos los miembros de esa clase.

Finalmente, la recuperación de datos ocupa un lenguaje de consulta artificial, con un glosario y sintaxis limitada, se hace una consulta completa de lo que se está buscando, en cambio la recuperación de información la consulta puede estar incompleta, debido a que ocupa un lenguaje natural, esto se da a que en la primera se está buscando documentos relevantes en lugar de fragmentos que coincidan con los elementos buscados. Por lo tanto la recuperación de datos es más sensible al error, porque puede darse por la falta de coincidencia sin recuperar los elementos deseados, los errores no afectan en gran escala al funcionamiento de la recuperación de información.

1.2.2.1. Tareas de recuperación de información

En la actualidad, existen más sistemas de recuperación de información, que permiten identificar los siguientes tipos de tareas de acuerdo a su naturaleza:

- **Recuperación de Ad-hoc:** Es la más representativa. En ella se basan los buscadores web, mediante las consultas los usuarios llegan al sistema de forma continua, mientras que los conjuntos de documentos permanecen sin movimiento en la colección del buscador y se encuentran de una manera estática, pero las preguntas de los usuarios son cambiantes y poseen un carácter dinámico. (Vilares Ferro, 2005)
- **Clasificación de documentos:** La recuperación ad-hoc es una forma de clasificar documentos en relevantes y no relevantes por cada consulta realizada. La clasificación

consiste en asignar un documento a una o más clases de documentos fijadas anteriormente en función a su contenido. (Vilares Ferro, 2005)

- **Clustering de documentos:** Su objetivo es generar una serie de clases o clúster a partir de un grupo de documentos dados previamente. Los clúster se atienen a los principios de maximización de la similaridad intra-clúster y de la minimización de la misma similaridad. (Vilares Ferro, 2005)
- **Segmentación de documentos:** Es la división del documento en partes más pequeñas y coherentes, es decir, fragmenta un documento en varios subtemas diferentes y se procesan de forma independiente como si fueran documentos separados. (Vilares Ferro, 2005)

1.2.2.2. Sistemas de recuperación de información

En la figura 1.3, se indican 3 componentes: entrada, procesamiento y salida.

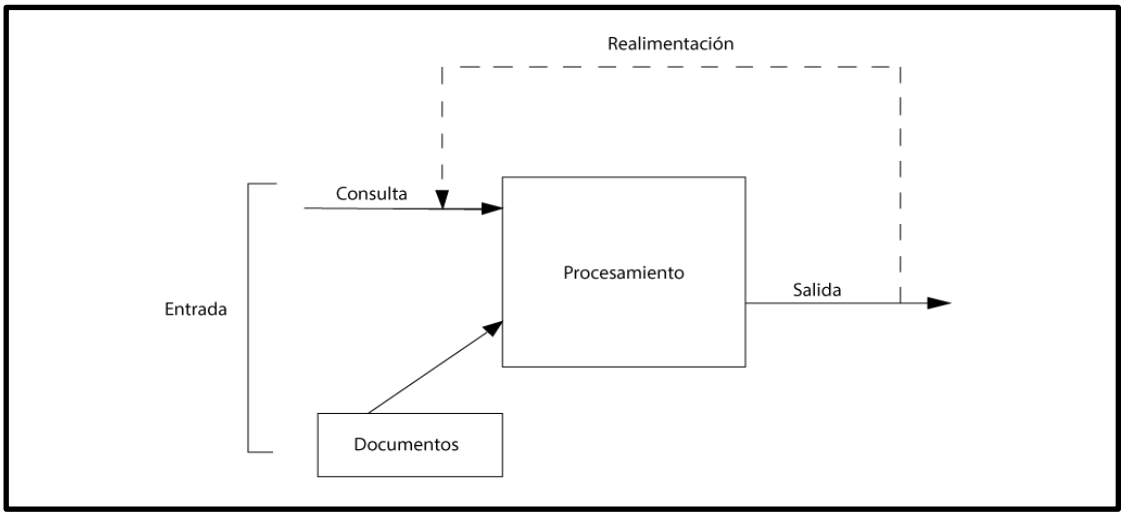

Figura 1-3. Sistema de recuperación de información
Fuente: (van Rijsbergen, 1979)

Iniciando con la ”Entrada” de información. El principal inconveniente es obtener la representación de cada documento, después de realizar la consulta desde un dispositivo. Por lo general, estos sistemas almacenan solo una representación del documento. Por ejemplo la representación puede ser un conjunto de palabras extraídas que se consideran significantes. Pero sería más satisfactorio que el dispositivo procese el lenguaje natural.

La parte denominada “Procesamiento”, realiza el trabajo de obtención de la información.

Para finalmente llegar a la “Salida”. Los sistemas de recuperación de información tienen como objetivo una búsqueda, el resultado del procesamiento de la consulta es una salida simple. La salida radica en el número de documentos recuperados y sus citas bibliográficas.

1.2.2.3. Extracción de información

(Diéguez, 2008) Es una forma de recuperar información que permite extraer automáticamente datos estructurados o semiestructurados desde documentos que pueden ser leídos en cualquier dispositivo. Este tipo de recuperación de información aumenta porque la información no se encuentra estructurada. No contienen metadatos asociados, siendo más accesibles sí, estos datos se representan como etiquetas XML[1].

Hoy en día la extracción de información se enfoca en tipos específicos de texto y no se obtiene buenos resultados en forma general.

1.2.2.4. Minería de texto

Existe confusión entre minería de datos y minería de texto, porque, los primeros son datos obtenidos directamente de una base de datos, que son transformados en conocimiento con ayuda de especialistas.

En cambio (Tan, 1999) dice: La minería de texto se divide en dos etapas principales, la primera de pre procesamiento y luego, una etapa de descubrimiento. La primera etapa cumple la función de manipular datos, para que estos puedan ser usados de una mejor forma y la segunda etapa es donde se generan nuevo conocimiento.

(Brun & Senso, 2004) exponen dos definiciones, la primera define a la minería de texto, como cualquier operación realizada para extraer y analizar texto de distintas fuentes con el objetivo de obtener inteligencia. La segunda la define como el descubrimiento de información y conocimiento que se desconocía, a partir de corpus textuales.

Además, la minería de texto, permite descubrir cantidades significativas de textos para analizar el conocimiento, que no se encuentra plasmado en los documentos, por lo que se puede decir que la minería de texto, es el proceso de derivación de nueva información.

Con el incremento de información en los últimos años, se ha hecho posible que se gestionen de mejor manera, con la ayuda de cuatro factores claves:

- Disminución del costo de los sistemas de almacenamiento.

[1] Extensible Markup Language: es un formato de texto simple, diseñado para cumplir con los retos de la publicación a gran escala, además juega un papel importante en el intercambio de una gran variedad de datos en la web (W3C, 2015).

- El incremento de velocidad de los procesadores.
- La mejora de confiabilidad y aumento de velocidad en la transmisión de datos.
- El desarrollo de mejores administradores de base de datos.

(Equihua, 2014) presenta tres actividades fundamentales que comprende la minería de texto:

1. La recuperación de información.
2. La extracción de información.
3. Encontrar relaciones entre los textos.

En pocas palabras, la minería de texto puede ayudar a la información implícita en los documentos más explícitos, lo cual permite ahorrar mucho tiempo. Esta nueva técnica es cambiante, se puede adaptar a muchas circunstancias y a diferentes situaciones por lo cual no existe ningún método a seguir. Pero, se pueden adoptar los pasos o etapas propuestos por (Microsoft, 2015), que podrían ser los fundamentales.

1. Determinar los objetivos.
2. Selección de datos o análisis de los mismos.
3. Determinación del modelo a ocupar.
4. Analizar los resultados.

1.2.3. Procesamiento de lenguaje natural

El Procesamiento de Lenguaje Natural nace en 1960, como una subárea de la Inteligencia Artificial y la Lingüística, con la finalidad de estudiar los problemas derivados del lenguaje natural.

En un principio tuvo una gran aceptación y éxito pero cuando se lo llevó a la práctica en campos no controlados y con vocabularios genéricos, empezaron a ocurrir problemas generados por la falta de comprensión de las máquinas del lenguaje natural.

(Covington, 1994) El Procesamiento de lenguaje natural es el uso de computadoras para entender lenguajes humanos (naturales) como inglés, francés y japonés, esto no quiere decir que el computador tenga pensamientos, sentimientos y conocimientos humanizados, sino que el dispositivo pueda reconocer y usar información expresada en lenguaje humano.

El PLN se ocupa en algunos campos como los expresados a continuación:

- Lingüística
- Ciencias de la computación
- Análisis Lingüístico

- Lenguaje
- Lenguaje Formal
- Comprensión del lenguaje
- Generación de textos
- Gramáticas Formales
- Matemática
- Neurociencia
- Definiciones empleadas en las gramáticas formales

La lingüística ha aportado grandes conocimientos sobre las lenguas naturales, las que se pueden estructurar en algunos niveles, como se indica en la tabla 1-2:

Tabla 1-2. Niveles de conocimiento

	Características del nivel de conocimiento lingüístico	
Nivel	**Declarativo**	**Procedural**
Fonológico	Sonidos hablados	Formar morfemas
Morfológico	Unidades de las palabras, Palabras	Formar palabras, Derivar unidades de Significado.
Sintáctico	Funciones estructurales de palabras	Formar oraciones
Semántico	Significado independiente del contexto	Derivar significado de oraciones
Discurso	Funciones estructurales de oraciones	Formar diálogos
Pragmático	Significado dependiente del contexto	Derivar significado de oraciones relativo al discurso circundante

Fuente: (Manaris & Slator, 1996)

(Covington, 1994) Brinda una organización en niveles sobre el conocimiento lingüístico tales como:

- **Nivel Fonológico:** estudia como los sonidos son usados en el lenguaje, cada lenguaje posee sus fonemas. El nivel fonológico estudia: las realizaciones acústicas, por lo que solo aparece en los sistemas de reconocimiento de audio. Un poco separado del procesamiento del lenguaje natural ya que este analiza la onda del sonido.
- **Nivel Morfológico:** la morfología se encarga de la descripción de la estructura que posee cada palabra y de los procesos que la integran. Existen tres procesos para la formación de

palabras que son: inflexión, derivación y composición, que son para evitar la expansión innecesaria de las palabras.

- **Nivel Sintáctico:** ofrece la construcción de las oraciones; Es el componente básico de los sistemas PLN, que ayuda a reconocer las oraciones gramaticales y a asignarles una estructura. En 1957 Noam Chomsky fue el primero en hablar sobre esto.
- **Nivel Semántico:** la semántica ofrece el significado de la frase o su posible significado, acercando esta al lenguaje, analizándolas independientemente del contexto que posea.
- **Nivel Pragmático:** es el uso que se le da al contexto o al significado más allá de lo que dice la frase. Ayuda a comprender información que se encuentra sobreentendida, pero que no se llega a expresar en las frases u oraciones.
- **Nivel Discurso:** se almacena el conocimiento, que logra relacionar el significado de las oraciones aisladas e integrarlas, para formar una unidad más grande. Este conocimiento se ocupa de interpretar los pronombres anafóricos, etc. y es necesario para que en los sistemas exista conocimiento del contexto.

Una representación de la estructuración en niveles se observa en la figura 1-4.

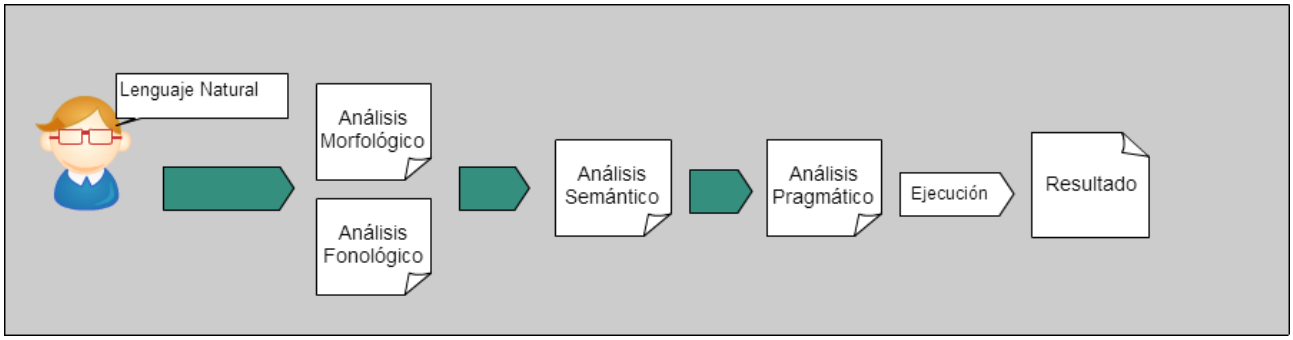

Figura 1-4. Niveles del conocimiento lingüístico
Elaboración: El autor de la tesis

Para realizar actividades de procesamiento de lenguaje natural, primero se debe normalizar el texto de la siguiente manera:

- Tokenización de palabras.
- Normalización de la estructura de las palabras.
- Tokenización de las oraciones.

Antes de continuar con la tokenización o segmentación se debe revisar algunos conceptos como:

- **Lema:** forma de citar de una palabra. Un ejemplo: el lema de “leíamos” es “leer”
- **Forma de la palabra:** forma completa de una palabra.

- **Tipo:** clase de elementos o elemento de un vocabulario.
- **Token:** instancia de un tipo de texto dado.

En primer lugar está la tokenización responsable de dividir el texto de entrada en oraciones y palabras. La forma más sencilla de tokenizar es separar los caracteres alfabéticos en función al carácter del espacio.

Los errores que ocurren con la tokenización varían dependiendo del idioma, por ejemplo el alemán porque las palabras compuestas se pueden escribir pegadas, sin ningún espacio.

Seguidamente, tenemos la normalización de palabras, permite agrupar a las que poseen un mismo significado, por ejemplo: juego, juegos y Juegos, se tiene que hacer que todas estas palabras se representen por una sola que es juego.

Una de las acciones que se realiza en la normalización, es cambiar todo a minúsculas, pero en el idioma español existen algunas excepciones; las cuales se pueden solventar con la lematización[2].

Finalmente, se realiza la segmentación de las oraciones de un texto, analizando cada elemento individualmente; localizando algunos símbolos especiales como: ".", "!" o "?", éstos identifican el fin de una oración, pero no siempre expresan ese significado. Para identificar cuando es el fin de una oración se usa un árbol de decisión, como el que se ilustra en la figura 1-5:

[2] En un diccionario o repertorio léxico, elegir convencionalmente una forma para remitir a ella todas las de su misma familia por razones de economía. (Real Academia Española, 2015)

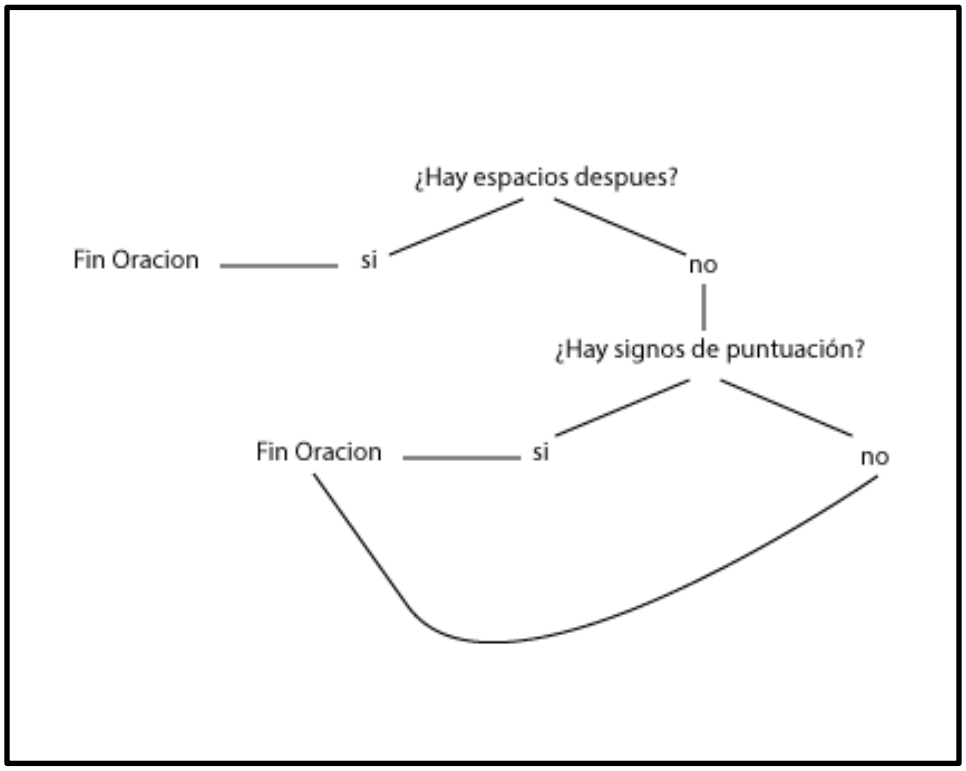

Figura 1-5. Árbol de decisión
Elaboración: El autor de la tesis

Un árbol de decisión no es más que un conjunto de "if" y "else" anidados, los cuales permiten escoger lo que se evaluará en cada nodo. Sin embargo, las oraciones no tienen que ser evaluadas por un árbol de decisión, se puede ocupar otros clasificadores como las redes neuronales, la regresión logística, entre otros.

1.2.3.1. Ambigüedad

El principal problema que posee el lenguaje natural es el análisis de la ambigüedad, en el nivel morfológico una palabra, podría tener muchas etiquetas.

En el nivel sintáctico, se considera ambiguo cuando se pueden asociar dos o más estructuras sintagmáticas[3], que son correctas a una frase. En este nivel, la palabra puede tener varios significados o se le puede dar diferentes sentidos.

1.2.3.2. Conocimiento lingüístico (CL)

(López García & Gallardo Paúls, 2011) Mencionan que el conocimiento lingüístico es una disciplina que se divide básicamente en lenguajes naturales y de máquinas. Siempre hay que enfrentarse a un objeto de estudio junto a la delimitación de las terminologías de las ciencias. La

[3] Se aplica a la relación que mantiene una palabra con las otras del mismo sintagma u oración (Real Academia Española, 2015).

lingüística computacional es equivalente al PLN, pero diferente de la lingüística informática y la ingeniería lingüística.

Se puede decir que CL es equivalente al PLN porque ambos poseen el mismo fin, (Grishman, 1986) explica que la lingüística computacional es el estudio de los sistemas de computación, utilizados para la comprensión y la generación de las lenguas naturales; comparándola con la definición de (Allen, 1995) sobre el PLN: El objetivo de una investigación es crear modelos computacionales del lenguaje lo suficientemente detallados que permitan escribir programas informáticos que realicen diferentes tareas en donde interviene el lenguaje natural.

Además, (López García & Gallardo Paúls, 2011) manifiestan que la lingüística computacional es la construcción de sistemas informáticos que puedan procesar estructuras lingüísticas, siendo su objetivo emular la capacidad lingüística humana, sin importar que su uso sea comercial o educativo.

Es importante conocer las principales aplicaciones prácticas de esta disciplina, las cuales (Moreno Sandoval, 1998) presenta una clasificación:

- Sistemas que emulan la capacidad humana de procesar el lenguaje natural, en este grupo lo que más se ocupa, son los traductores automáticos, la extracción y recuperación de información e interfaces hombre-máquina.
- Sistemas de ayuda a tareas lingüísticas, este está conformado por herramientas ocupadas por lingüistas para facilitarles su trabajo.
- Programas de escritura y composición textual, estas aplicaciones han sido desarrolladas para que se familiaricen con el usuario.
- Enseñanza asistida, incluyen los programas educativos para la aprendizaje de idiomas.

1.2.3.3. Modelos

Los modelos permiten hacer inferencias acerca de un objeto modelado.

(López García & Gallardo Paúls, 2011) Refieren que los fenómenos lingüísticos que suelen tener una gran variedad de modelos matemáticos asociados y éstos proporcionan conocimiento parcial sobre un fenómeno específico, por lo tanto se tiene que escoger el modelo más adecuado para cada caso. Los modelos se clasifican en simbólicos, estadísticos e híbridos. Los simbólicos, emplean reglas y algoritmos que operan con las estructuras de datos, que representan el conocimiento de lenguaje natural; el estadístico, en cambio, involucra el corpus o pequeñas colecciones de una muestra del lenguaje, siendo etiquetados y usados para crear modelos

estadísticos. Finalmente, las híbridas son una combinación de los dos modelos descritos anteriormente, con la finalidad de tener las ventajas de cada modelo.

1.2.3.4. Métodos de aprendizaje

Métodos estadísticos que se implementaron al Procesamiento de Lenguaje Natural, estos se dividen en dos grandes grupos, dependiendo de la información que se necesita para la aplicación:

- **Métodos supervisados:** requieren una gran cantidad de datos de entrenamiento para aprender a partir de ellos, primero procesan datos, luego aplican lo aprendido a los nuevos datos ingresados. Los resultados de este método, dependen de la calidad del archivo de entrenamiento, en función de la cantidad de los mismos y su representatividad del lenguaje, en el problema que se desea resolver (Ortega Rodríguez, 2008).
- **Métodos no supervisados:** no necesitan de archivos de entrenamiento, lo único que ocupan son los datos de entrada, por lo tanto, no poseen una fase de entrenamiento, simplemente procesan los datos de entrada. Poseen algunas ventajas sobre los métodos supervisados, como la velocidad de respuesta, pues no necesita un archivo de entrenamiento, pero puede ser un problema cuando se necesita realizar tareas referentes a un idioma (Ortega Rodríguez, 2008).

En este proyecto de investigación, se utiliza métodos supervisados. En PLN los archivos de entrenamiento se denominan corpus, que son una gran colección de textos, en ellos puede haber información adicional como un conjunto de palabras. Por ejemplo, puede contener la categoría gramatical de cada palabra.

Los métodos supervisados luego de procesar el corpus, intentan predecir un resultado para una determinada tarea, y así puede realizar un procesado preciso sobre un nuevo valor de entrada.

Existen algunas técnicas estadística aplicadas al PLN como: Los modelos de Markov, los modelos de aprendizaje basados en memoria, los modelos de máxima entropía, las máquinas de soporte vectorial por nombrar algunos.

1.3. Nivel Sintáctico

En el apartado 1.2.3 se mencionó al nivel sintáctico. En este nivel se combinan las palabras mediante reglas, las cuales permiten generar construcciones gramaticalmente correctas. Además, se tiene que generar la estructura de las categorías sintácticas de las unidades léxicas, que se encuentran en la oración.

1.3.1. Conceptos y gramática

(Martí & Taulé, 2011) mencionan que un analizador sintáctico tiene como objetivo establecer las relaciones estructurales y de dependencia entre las palabras dentro de la frase. El análisis consiste básicamente, en la identificación de sintagmas y anotaciones con sus funciones correspondientes. Se trata de un recurso que, aunque obtiene resultados con un nivel de acierto respetable, todavía no se considera resuelto.

La gramática está formada por un conjunto de reglas que se observan en la tabla 1-3:

Tabla 1-3. Reglas Gramaticales

Reglas Gramaticales
O --> SN, SV
SN --> Det, N
SN --> Nombre Propio
SV --> V, SN
SV --> V
SP --> Preposición, SN
SN = sintagma nominal
SV = sintagma verbal
SP = sintagma preposición
N = núcleo
O = oración
Det = determinante

Fuente: (Argomedo Pflücker & Cóndor Ruiz, 2014)

Este resultado, también se puede representar en forma de árbol binario, en vista que los árboles son utilizados para representar de forma gráfica la estructura de una oración.

1.3.2. Jerarquía de Chomsky

(Gallego, 2008) Dependiendo de las reglas de producción se puede definir la complejidad del lenguaje, originando una clasificación en función a las reglas de producción. Chomsky propone una jerarquía de cuatro clases que son:

- Gramáticas regulares.
- Gramáticas libre de contexto.
- Gramáticas dependientes del contexto.
- Gramáticas con estructura de fase.

La lengua natural puede estar situada entre los lenguajes dependientes de contexto y los no dependientes de contexto. Esta clasificación no es la única que existe.

1.3.3. Análisis Sintáctico

Intenta determinar la validez que posee una expresión gramatical y su estructura sintáctica a través de un análisis. Este proceso da como resultado un árbol sintáctico, que representa la estructura gramatical que pertenece a una frase.

1.3.4. Análisis Sintáctico Superficial

Basado en fragmentos (chunks).

(Galicia Haro & Gelbukh, 2007) mencionan: cuando se lee un pedazo de frase, esos trozos corresponden de alguna manera a los patrones prosódicos. Analizando las partes de la oración, el estudio de estos pedazos, son la base del análisis sintáctico total que determina los verbos y los grupos de preposiciones.

1.4. WordNet

("About WordNet," 2015) es una base de datos léxica en el idioma inglés, con sustantivos, verbos, adjetivos y adverbios, que se agrupan en conjuntos de sinónimos cognitivos (synsets), cada uno expresando un concepto distinto. Los synsets están vinculados entre sí por medio de relaciones conceptuales ya sean semánticas o léxicas.

WordNet ofrece un buscador que se asemeja a un diccionario de sinónimos, pero posee algunas diferencias significativas en las palabras, no solo se encuentran interrelacionadas en formularios sino que también le dan el sentido específico a la misma. Cabe resaltar que WordNet etiqueta las relaciones entre las palabras, que en un diccionario no siguen ningún patrón.

La explicación anterior enfatiza, que uno de los puntos fuertes del WordNet es la sinonimia[4]. Los sinónimos son palabras que tienen el mismo significado y son intercambiables en varios contextos. Los synsets poseen una breve descripción, compuesta por una palabra o un grupo de las mismas.

("About WordNet," 2015) indica algunos tipos de relaciones entre synsets que se explican a continuación:

- Super-Subordinate (Hiponimia). Relación que vincula synsets a unos más específicos, es una relación transitiva.
- Part-Whole (Meronimia). Indica la relación de herencia que existen en los synsets.

WordNet realiza una distinción entre nombres comunes e instancias, las instancias serán siempre las hojas terminales de sus jerarquías.

[4] Coincidencia de significados entre dos o más vocablos. (Real Academia Española, 2015)

Los verbos se organizan en jerarquías, que se encuentran en la parte inferior del árbol, expresando características específicas de un evento.

Los adjetivos, en cambio se encuentran organizados en términos de antonimia[5], como feliz – triste, son antónimos directos que reflejan un fuerte contrato semántico de sus miembros.

Como no hay muchos adverbios en WordNet, la mayoría se derivan de forma directa de los adjetivos, a través de afijación morfológica en la lengua inglesa.

1.5. EuroWordNet

Al igual que WordNet, EuroWordNet es una base de datos multilingüe con wordnets sobre varios idiomas europeos como: holandés, italiano, español, alemán, francés, checo y estonio. Están estructuradas de la misma forma que el americano. Estos wordnets representan un sistema de lenguaje único, se encuentran vinculadas a través de un índex al WordNet americano, y gracias a esta interconexión es posible buscar palabras parecidas en otro idioma.

EuroWordNet tiene el enfoque de construir wordnets principalmente de recursos existentes. Así, tiene a posibilidad de combinar y comparar la información de múltiples recursos creados de forma independiente, estas comparaciones permiten ver la consistencia y calidad que tienen los recursos.

A diferencia del WordNet original, la mayoría de los otros WordNets no están disponibles de forma gratuita.

1.6. Normalización de corpus

1.6.1. Lingüística

El corpus es una fuente de información lingüística, con colecciones de documentos con fines específicos. Representan el lenguaje que se va a analizar e indica que los criterios de selección del corpus se mantengan a lo largo del análisis. De esta forma es posible acceder a una porción que posee las mismas características.

(Torruella & Llisterri, 1999) Proporcionan pautas a considerar sobre qué es y no es un corpus en la siguiente clasificación de recopilación de textos:

- Archivos o colección informática de textos: colección de texto que no poseen alguna relación entre ellos.

[5] Oposición de dos términos de significado contrario (Real Academia Española, 2015).

- Biblioteca de textos electrónicos: colección de textos que poseen un formato siguiendo algunos estándares rigurosos.
- Corpus informatizado: es una recopilación de textos, que fueron seleccionados de acuerdo a criterios lingüísticos, con el fin de reflejar el comportamiento de una o más lenguas.

(Alcántara Plá, 2007) Ofrece otro tipo de clasificación de los corpus relacionados con el tipo de documento que estos manejan:

- Corpus especializados: se componen por textos elegidos que poseen características fijas. Por ejemplo: corpus temáticos y los de registro.
- Corpus generales: son generales y de aspectos muy básicos. Un ejemplo claro es: Corpus de Referencia del Español Actual (CREA, http://corpus.rae.es/creanet.html)
- Corpus comparables: se componen por varios subcorpus que comparten características básicas excluyendo el idioma. Ejemplo: corpus de enseñanza de las lenguas.
- Corpus paralelos: compuesto por subcorpus de textos idénticos pero de diferentes idiomas. Ejemplo: textos de legislación.
- Corpus históricos: son texto de distintos periódicos históricos que se ocupan para estudios diacrónicos. Ejemplo: corpus de Helsinki, que recolecta información desde el año 700 hasta el 1700.

1.6.2. Etiquetado del corpus

Para el análisis del corpus es necesario su etiquetación, porque permite escoger almacenar y recuperar información. Para esto se ocupa un lenguaje de anotaciones, que es el XML, lenguaje desarrollado por la W3C.

Citamos algunas características que tiene XML que lo han convertido en el estándar para el etiquetamiento de recursos lingüísticos:

- Permite una configuración libre sin ninguna restricción del etiquetado del corpus. Compatible con el estándar UNICODE, que posibilite la marcación de distintos idiomas.
- Ayuda a conservar la estructura de los documentos, lo que le permite realizar un intercambio de información.
- Es independiente, por lo tanto se puede mantener, publicar y editar en diferentes medios.
- Es un lenguaje abierto y gratuito.

(Alcántara Plá, 2007) Señala una descripción de las herramientas informáticas ocupadas para el etiquetamiento.

- Asistente etiquetador: ayuda durante el etiquetamiento con relación al tipo de marcado que se necesite.
- Sistemas de etiquetado automático: no necesitan supervisión humana para realizar el etiquetado, estos analizan un documento y lo procesan con las etiquetas respectivas.
- Sistemas de etiquetado semiautomático: la supervisión humana ayuda a completar el etiquetado, el lingüista revisa y corrige los resultados.

1.6.3. Selección de etiquetas

Se distinguen algunos tipos de etiquetamiento:

- Anotaciones sintácticas: marcación estructural de párrafos, oraciones y frases.
- Lematización: anotación de lemas que están en el texto para luego ser utilizados en análisis léxicos.
- Anotaciones morfosintácticas: componentes morfosintácticos de la lengua.
- Anotaciones semánticas: relaciones de carácter semántico entre los elementos de la oración o clases semánticas de las palabras de texto.
- Anotaciones pragmáticas y discursivas: analizan los actos comunicativos del lenguaje.
- Anotaciones prosódicas: etiquetan al lenguaje hablado.

1.6.4. Criterios generales para la selección de corpus

(Pérez Hernández, 2002) menciona cuatro aspectos generales, necesarios para la selección de corpus.

- **Cantidad.** Hay que tener en cuenta que un corpus debe ser amplio, porque ofrece una gran cantidad de texto. No es necesario tener un límite a la cantidad de texto que se recopile, pero este deben ser completo, porque omitir una parte de un texto puede ocasionar resultados no esperado.
- **Calidad.** Se encarga de asegurar que en cualquier corpus de grandes proporciones se represente un espectro amplio del uso del lenguaje. Cuando no se cumple esta condición el corpus tendra una calidad baja y no se lograrian los resultados deseados. La calidad se encuentra limitada por lo real que puedan ser estos corpus más que por su variedad.
- **Simplicidad.** Facilita la recuperación de la información a los especialistas y ordenadores. El etiquetado que poseen los textos ya sean de datos específicos o generales componen la información que llamamos metadata, estos datos permiten realizar búsquedas más precisas en un corpus o en Internet, por lo tanto la simplicidad es necesaria en la recopilación de corpus especializados permitiendo seleccionar textos con mayor rapidez.

- **Documentación.** Está relacionada ampliamente con la simplicidad. La documentación permite agregar o quitar datos del texto original y ubicarlos con mayor rapidez a los textos. De acuerdo a las especificaciones dadas por los EAGLES, la información tiene que estar separada del cuerpo para tener una búsqueda más rápida y eficiente.

1.7. Etiquetado Gramatical (Part-of-Speech Tags)

El término "part of speech tagging" o en su forma corta POS tagging, según (Indurkhya & Damerau, 2010), el etiquetado permite en gran parte las tareas del procesamiento de lenguaje natural, el mismo es aplicado como un compromiso razonable entre la precisión y la utilidad. Además, proporciona una información estructural de nivel superior y la relación que existe entre las palabras. También realiza el etiquetado de forma más rápida y precisa que el análisis. Los buenos etiquetados logran desarrollar un dominio mucho más rápido que los mejores análisis sintácticos.

(Mitkov, 2005) El etiquetado gramatical ha sido reconocido en la lingüística por un largo tiempo, además se ha podido distinguir ocho clases de palabras que usan un criterio formal: Sustantivo, verbo, participio, artículo, pronombre, preposición, adverbio, conjunción. El etiquetado gramatical puede aparecer en cualquier lenguaje natural y los criterios para este son gramaticales en lugar de semánticos: distribución sintáctica, función sintáctica y las clases morfosintácticas y sintácticas que pueden ser asignadas al etiquetado gramatical.

Por lo tanto, se puede decir que es otra forma de asignación directa de descriptores, etiquetas o tokens de entrada.

Algunas de las tareas para las cuales el etiquetado ha sido útil son: la traducción automática, la extracción de información, la recuperación de información y el procesamiento sintáctico de nivel superior. Además, se destacan algunos aspectos para una comprensión, por ejemplo en la traducción automática existe la posibilidad de que una palabra en el idioma de origen traducida al idioma de destino sea dependiente del etiquetado de esa palabra, pero si se sabe que el registro es un sustantivo la traducción de la palabra es casi segura.

Con la extracción de información tenemos los patrones para obtener cierta información, que hacen referencia al etiquetado gramatical o POS tagger.

1.8. Clasificación de emociones

(Cabral Morales, 2006) dice, el psicólogo Robert Plutchik identifico y clasifico las emociones que experimentan los seres humanos y animales, en 8 categorías básicas que motivan la conducta

adaptativa, estas emociones son: Temor, Sorpresa, Tristeza, Repugnancia, Enojo, Esperanza, Dicha y Aceptación. Esta no es la única clasificación que existe.

Es importante diferenciar que es un sentimiento y una emoción. Las emociones son respuestas que se dan a determinadas situaciones y se originan gracias a un impulso externo.

En cambio, el sentimiento es un término que abarca más que solo sentir el estímulo. (Oatley, 1992) lo define como: una experiencia afectiva en cierta medida agradable o desagradable, que supone una cualidad fenomenológica característica, que comprende tres sistemas de respuesta: cognitivo y subjetivo, conductual y expresivo y fisiológico y adaptativo.

1.9. Análisis de Herramientas

1.9.1. Amazon Mechanical Turk

Amazon Mechanical Turk es un servicio que ofrece Amazon, que permite realizar trabajos simples y de bajo costo, requieren de inteligencia humana en vista que la máquina no puede resolver ciertas tareas. Esta herramienta sirve de consulta de tareas para los usuarios.

(Amazon, 2014) Tiene como objetivo que la inteligencia humana sea sencillo, escalable y rentable. Los aspectos relevantes de este servicio son:

- Personal bajo demanda. Proporcionan trabajadores que pueden ayudar a completar el trabajo cuando y donde se los necesite.
- Estructura de coste bajo. Se reducen los costes gracias a que el personal es bajo en demanda.
- Determinación de precio. Las empresas tienen libertad para definir el precio con el fin de atraer un buen número de empleados.
- Cualificación del personal. Se realizan pruebas rápidas al personal antes de que estos puedan trabajar en sus tareas.

Además, es de gran utilidad para los desarrolladores porque pueden realizar distintas aplicaciones como, experimentaciones en las redes sociales, búsquedas de personas desaparecidas, etc.

CAPÍTULO II

ANÁLISIS DE LAS HERRAMIENTAS

2.1. Introducción

Este capítulo se describe y analiza algunas herramientas para el Procesamiento de Lenguaje Natural y para la extracción de información, que es una de las principales funciones que realiza el PLN, además se seleccionará la herramienta que mejor se adapte al objetivo del proyecto, primero se presentara una explicación de estas herramientas su operatividad y funcionamiento.

2.2. Stanford CoreNLP[6]

CoreNLP es un conjunto de herramientas para el procesamiento del lenguaje natural que se basa en modelos estadísticos, desarrollado por el Grupo de Procesamiento de Lenguaje Natural de la Universidad de Stanford, proporciona un alto nivel de PLN, desarrolladas en Java, pero también existe para las siguientes plataformas: Apache Thrift, C#/F#/.NET, Ruby, Python, Perl, Scala, Clojure, ZeroMq Server y Javascript, estos lenguajes pueden variar dependiendo del módulo desarrollado.

En este kit de herramientas de análisis del lenguaje natural se puede ingresar un texto sin procesar y de esta forma llegar a presentar la estructura base de las palabras, partes de la oración, nombres de personas u organizaciones, etc. Una de las finalidades que tiene es facilitar la aplicación de herramientas de análisis lingüístico a una parte de un texto.

La herramienta integra algunas aplicaciones de PLN las cuales son las siguientes:

- Etiquetador (POS)
- Reconocedor de la entidad (NER)
- El analizador
- El sistema de resolución de la correferencia
- El análisis de sentimientos
- El patrón de aprendizaje bootstrapper

Esta herramienta ofrece un análisis para el texto en inglés compatible con los idiomas que se observan en la tabla 2-1:

[6] https://opennlp.apache.org/

Tabla 2-1. Servicios disponibles para los lenguajes Soportados CoreNLP

Anotador	Arábigo	Chino Mandarín	Inglés	Francés	Alemán	Español
Tokenizador	✓	✓	✓	✓	✓	✓
División de sentencias	✓	✓	✓	✓	✓	✓
Truecase			✓			
POS	✓	✓	✓	✓	✓	✓
Lema			✓			
Gender			✓			
NER		✓	✓		✓	✓
Regexner	✓	✓	✓	✓	✓	✓
Parsing	✓	✓	✓	✓	✓	✓
Análisis de dependencias		✓	✓			
Análisis de sentimientos			✓			
Coref			✓			

Fuente: (Manning et al., 2014)

Además se puede encontrar los archivos empaquetados de los modelos para algunos idiomas.

Los modelos a usar son los disponibles para tratar los textos en español, que son: tagger (etiquetador), parser (analizador) y ner (reconocedor de entidades). Estos permitirán obtener la estructura gramatical de los textos. A continuación un ejemplo de su funcionamiento.

2.2.1. POS Tagger

Es una parte del software que permite leer el texto de un idioma y asignar las partes de una oración, en cada una de las palabras del texto como: verbo, adverbio, sustantivo u otra, es decir, permite etiquetar las palabras de una cadena y asigna el resultado dependiendo de que se trate esa palabra.

Análisis de la siguiente frase:

"Esta es una oración de prueba"

Escogemos el IDE de NetBeans, el grupo de Stanford ofrece un grupo de modelos para diferentes idiomas, por defecto es el inglés, pero en este trabajo se usa el modelo que está en español.

Se realizan los siguientes pasos:

1. Crear un nuevo proyecto

2. En la dirección o ruta, donde se ubica el nuevo proyecto se crea una carpeta con el nombre de taggers (puede tener otro nombre)
3. Descomprimir los archivos obtenidos de la página de Stanford[7], ubicarlos en la carpeta creada en el paso anterior, estos archivos tienen la extensión tagger y props.
4. Importar la librería al IDE, para ocupar el modelo.
5. Luego de concluida la configuración, codificar el método principal en el que constará un constructor de la clase MaxentTagger, al cual se le tiene que enviar como parámetro un archivo entrenado.

```
//Iniciar el tagger
MaxentTagger tagger = new MaxentTagger("taggers/spanish.tagger");
```

Figura 2-1. Codificación: Ubicación del modelo tagger CoreNLP
Elaboración: El autor de la tesis

6. Por último se etiqueta la cadena que queremos procesar y presentamos el resultado:

```
// Ejemplo de una cadena
String ejemplo = "Esta es una oración de prueba";
// Cadena
String taggeo = tagger.tagString(ejemplo);
// Presentamos el resultado
System.out.println(taggeo);
```

Figura 2-2. Codificación: Taggeo y presentación CoreNPL
Elaboración: El autor de la tesis.

Al momento de ejecutar este código el resultado es el siguiente:

Esta_pd000000 es_vsip000 una_di0000 oración_nc0s000 de_sp000 prueba_nc0s000

Pero en este caso que significa "_pd000000", es una nomenclatura propuesta por el grupo EAGLES para la asignación de etiquetas.

Al ejemplo completo lo podemos encontrar en el anexo 1.

[7] http://nlp.stanford.edu/software/tagger.shtml

2.2.2. Reconocedor de la entidad (NER)

Es una subtarea de la extracción de información que se encarga de buscar y clasificar los elementos del texto en categorías ya definidas, estos pueden ser los nombres de personas, organizaciones, lugares, tiempo, etc.

Presentaremos un ejemplo de esta funcionalidad, con el proyecto creado anteriormente:

Para realizar este ejemplo utilizamos la siguiente frase:

"Soy un estudiante de la Universidad Técnica Particular de Loja, vivo en la ciudad de Loja"

Con esta frase, se realiza los siguientes pasos:

1. En el proyecto creamos una nueva clase con cualquier nombre, en este ejemplo se utiliza Ner.java
2. Luego procedemos a colocar las dependencias necesarias, el cual sería el .jar, en el que se encuentran los modelos en español (stanford-spanish-corenlp-2015-01-08-models.jar), en este caso utilizaremos el paquete de stanford (stanford-corenlp-3.5.0.jar), en el que está incluido el NER.
3. Codificamos el método principal, en el que se agrega la ruta en la que se encuentra el modelo de clasificación y se añaden las excepciones necesarias para su funcionamiento:

```
public class Ner {

    public static void main(String[] args) throws IOException, ClassCastException, ClassNotFoundException {
        AbstractSequenceClassifier<CoreLabel> classifier = CRFClassifier.getClassifier("edu/stanford/nlp/models/ner/spanish.ancora.distsim.s512.crf.ser.gz");
    }

}
```

Figura 2-3. Codificación: Ruta de modelo de clasificación Ner
Elaboración: El autor de la tesis

4. Se procede a ubicar la frase o texto a analizar y se envía al clasificador, para obtener algún resultado:

```
String ejemplo = "Soy un estudiante de la Universidad Técnica Particular de Loja, vivo en la ciudad de Loja";

List<List<CoreLabel>> out = classifier.classify(ejemplo);
```

Figura 2-4. Codificación: Texto a clasificar Ner
Elaboración: El autor de la tesis

5. Se obtiene las entidades de la cadena y se presenta el tipo de entidad al que pertenece cada token.

```
for (List<CoreLabel> oracion : out) {
    for (CoreLabel palabra : oracion) {
        System.out.print("('"+palabra.word() + "','" +palabra.get(CoreAnnotations.AnswerAnnotation.class) + ") ");
    }
    System.out.println();
}
```

Figura 2-5. Codificación: Presentación clasificación Ner
Elaboración: El autor de la tesis

6. Finalmente el resultado que presenta la ejecución de este ejemplo es la siguiente:

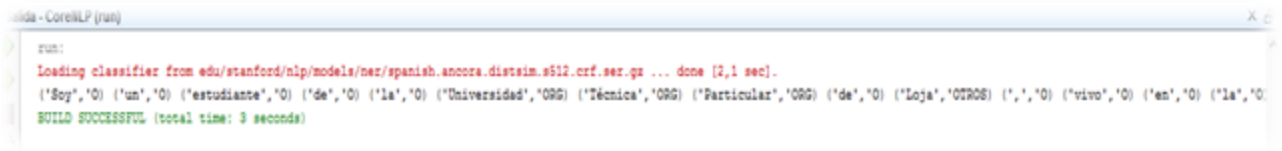

Figura 2-6. Resultado clasificación Ner
Elaboración: El autor de la tesis

En la figura 2-6, el resultado se presenta como tokens y la clasificación de los mismos, dependiendo de la categoría a la que pertenece por ejemplo Universidad Técnica Particular de Loja la ha clasificado de la siguiente manera:

('Universidad','ORG) ('Técnica','ORG) ('Particular','ORG) ('de','O) ('Loja','OTROS)

Se indica que hay una organización, además, presenta como clasifico a toda la frase. El modelo en español solo reconoce organizaciones, lugares y otros.

El ejemplo completo está en el anexo 2

2.2.3. Parser

(The Stanford Natural Language Processing Group, 2014), define al analizador como:

Programa que funciona con la estructura gramatical de las oraciones, como grupos de palabras que están juntas e identifica que palabras son el sujeto u objeto del verbo. En la década de los noventa el desarrollo de los analizadores estadísticos fue un gran avance, aunque siguen cometiendo errores, pero funcionan bien en la mayoría de los casos.

Como se observa en el siguiente ejemplo:

Se seguirá ocupando el mismo proyecto y se analizará la siguiente oración:

"El reino canta muy bien."

Con esta información se realiza lo siguiente:

1. Crear un nuevo paquete y una nueva clase con un nombre que los identifique.
2. Codificar el método principal y ubicar la dirección en donde se encuentra el modelo.

```
LexicalizedParser lexpars = LexicalizedParser.loadModel("edu/stanford/nlp/models/lexparser/spanishPCFG.ser.gz");
```

Figura 2-7. Codificación: Ruta modelo de CoreNLP parser
Elaboración: El autor de la tesis

3. Cargar el texto y hacer uso del tokenizador

```
String texto = "El reino canta muy bien.";

TokenizerFactory<CoreLabel> tokenizerFactory = PTBTokenizer.factory(new CoreLabelTokenFactory(), "");
Tokenizer<CoreLabel> token = tokenizerFactory.getTokenizer(new StringReader(texto));
List<CoreLabel> rawPalabras = token.tokenize();
```

Figura 2-8. Codificación: Tokenizar la oración CoreNLP parser
Elaboración: El autor de la tesis

4. A partir de los tokens se crea un árbol y se realiza la presentación del mismo.

```
Tree parse = lexpars.apply(rawPalabras);
parse.pennPrint();
```

Figura 2-9. Codificación: Creación y presentación del árbol coreNLP parser
Elaboración: El autor de la tesis

5. Finalmente presentamos el resultado que obtenemos de la ejecución de este ejemplo.

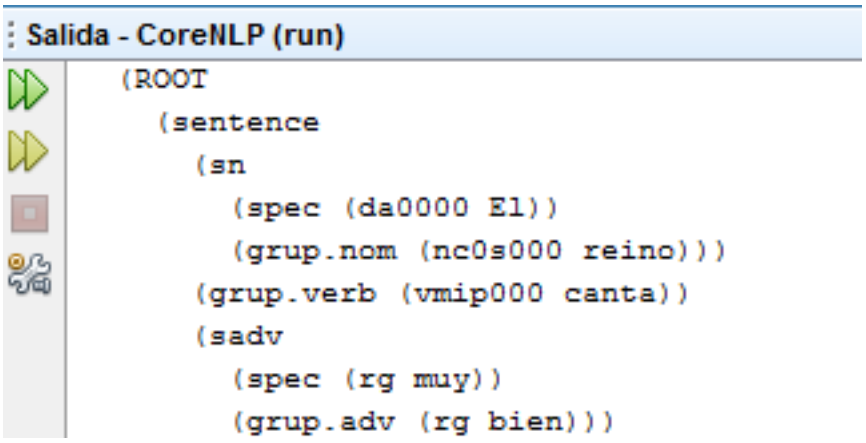

Figura 2-10. Resultado del árbol coreNLP parser
Elaboración: El autor de la tesis

El resultado en la figura 2-10, muestra la oración separada en grupos e identifica las relaciones gramaticales que éstas poseen.

El ejemplo completo se encuentra en el anexo 3

2.2.4. Sistema de resolución de la correferencia

Esta tarea consiste en identificar expresiones diferentes que aparecen en algún texto, explican o hablan del mismo objeto, está disponible para modelos en inglés. Para esta investigación se utilizará los modelos que permiten tratar los textos en español.

2.2.5. Análisis de sentimientos

CoreNLP ofrece un analizador de sentimientos enfocado en el idioma inglés, el análisis de sentimientos se lleva a cabo gracias a diccionarios de sentimientos, mediante la tokenización del texto para obtener calificaciones individuales de los tokens y agregar estos valores para la calificación final. Este módulo funciona a nivel de oración y no de documento.

2.3. OpenNLP[8]

Es una biblioteca desarrollada en Java por Apache OpenNLP, es compatible con las tareas que realiza el procesamiento del lenguaje natural, como detección de oraciones, tokenización, etiquetado y otras tareas. Está basado en el aprendizaje de máquina para procesar el lenguaje natural.

El paquete de OpenNLP posee las siguientes funcionalidades:

- Tokenización
- Detección de oraciones
- POS tagging
- Chunking
- Parsing
- NER
- Correferencias

OpenNLP ofrece funcionalidades en otros idiomas, que se indican en la tabla 2-2:

Tabla 2-2. Servicios disponibles para los lenguajes Soportados OpenNLP

Componente	Inglés	Danés	Alemán	Español	Holandés	Portugués	Sámi	Thai
Detección de oración	✓	✓	✓	✓	✓	✓	✓	✓
Tokenizador	✓	✓	✓	✓	✓	✓	✓	✓
NER	✓			✓	✓			

[8] http://nlp.stanford.edu/software/corenlp.shtml

Coref	✓							
Chuking	✓							
POS	✓	✓	✓	✓	✓	✓	✓	✓
Parsing	✓							

Elaboración: El autor de la tesis

Solo se ocupará las funcionalidades que se encuentran en español, y éstas son: POS tagging, tokenización, NER y detección de oraciones.

2.3.1. Detección de oraciones

Puede detectar el carácter que da finalización a la frase y también cuando no termina, (Apache OpenNLP Development Community, 2014) define como frase: un espacio en blanco más largo, recortada en secuencias entre dos signos de puntuación.

Para esto se indica un ejemplo del funcionamiento y se analizara la siguiente oración:

"En el día más brillante, en la noche más oscura,

El mal no escapará a mi vista.

Que aquellos que adoran al mal,

Teman mi poder: ¡LA LUZ DE LINTERNA VERDE!"

Igual que en la prueba anterior se ocupa el mismo IDE, porque trabaja con el mismo lenguaje de programación, pero antes de comenzar tenemos que descargar los modelos[9] correspondientes.

Luego procedemos con los siguientes pasos:

1. Crear un proyecto con el nombre que se desee:
2. Agregar la librería: opennlp-tools-1.5.0.jar, para usar los modelos que se añadirán a continuación. Seleccionar el proyecto y agregar la librería.
3. En la ruta de creación del proyecto, crear una carpeta con el nombre que se desee para luego colocar el modelo correspondiente.
4. Con los pasos anteriores se realiza un ejemplo, creando una clase con cualquier nombre (SentecesDetector), en su método principal se coloca la dirección o ruta donde se encuentra el modelo, en nuestro caso sería como se muestra en la figura 2-11.

[9] http://sourceforge.net/projects/opennlp/files/

```
String MODELDIR = "C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\sentdetector\\es-sent.bin";
SentenceDetector sendet = new opennlp.tools.lang.spanish.SentenceDetector(MODELDIR);
```

Figura 2-11. Codificación: Ruta del modelo OpenNLP
Elaboración: El autor de la tesis

5. Luego se realiza la detección de la oración del texto descrito con anterioridad, como consta en la figura 2-12:

```
String sentences[] = sendet.sentDetect(texto);
```

Figura 2-12. Codificación: Detección de la oración OpenNLP
Elaboración: El autor de la tesis

6. Finalmente se presenta un resultado como el de la figura 2-13:

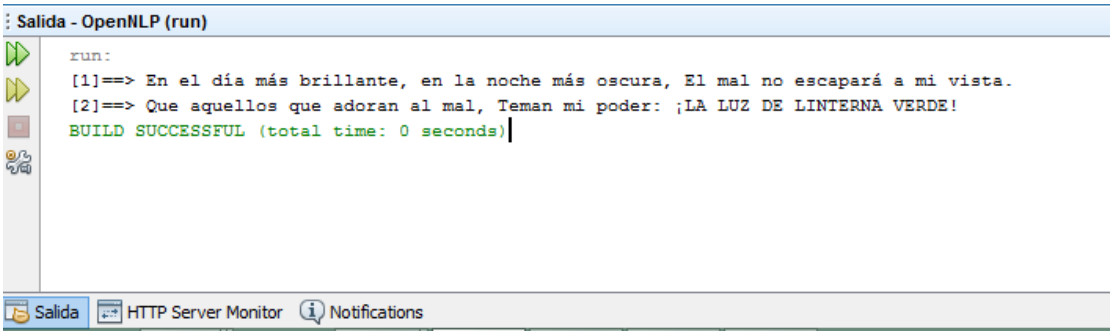

Figura 2-13. Resultado: Detección de oraciones OpenNLP
Elaboración: El autor de la tesis

En la figura 2-13, determina que con el texto ingresado se obtuvo dos oraciones, que se encuentran separadas o delimitadas por el punto.

Al utilizar la última versión de OpenNLP, esta no posee un modelo actual para la detección de oraciones, pero se pueden ocupar los modelos anteriores entrenándolos primero.

Nos guiamos de la documentación que nos proporciona el API[10], en el que se toma en cuenta tres puntos importantes, que se detalla a continuación:

1. Abrir un flujo de datos.

```
Charset charset = Charset.forName("UTF-8");
ObjectStream lineStream = new PlainTextByLineStream(new FileInputStream(RUTA_TEXTO), charset);
ObjectStream sampleStream = new SentenceSampleStream(lineStream);
```

Figura 2-14. Codificación: Abrir flujo de datos OpenNLP – Detección de oraciones
Elaboración: El autor de la tesis

2. Llamar al Método de entrenamiento “SentenceDetectorME.train”

[10] Abreviatura de Aplication Programming Interface. Un API no es más que una serie de servicios o funciones que el Sistema Operativo ofrece al programador, como por ejemplo, imprimir un carácter en pantalla, leer el teclado, escribir en un fichero de disco, etc.

```
model = SentenceDetectorME.train("es", sampleStream, true, null, TrainingParameters.defaultParams());
```

Figura 2-15. Codificación: Método SentenceDetectorME OpenNLP – Detección de oraciones
Elaboración: El autor de la tesis

3. Guardar los cambios realizados.

```
modelOut = new BufferedOutputStream(new FileOutputStream(MODELDIR));
model.serialize(modelOut);
```

Figura 2-16. Codificación: Guardar OpenNLP – Detección de oraciones
Elaboración: El autor de la tesis

Ejecutar y ocupar el modelo para la detección de oraciones ya entrenado y obtener el resultado que se mostró en la figura 2-13.

En archivo de entrenamiento, tiene una oración por línea, como se ve en la figura 2-17.

```
libro de audio editado por 3 uves dobles punto lee eme pe tres punto com.
El filósofo autodidacta.
Abentofail (1110-1185)
Filósofo y médico hispanoárabe.
Nació en Guadix (Granada) hacia el año 1110, fue médico del sultán almohade Abü Ya'qub Yusuf y ocupó un puest
En el núcleo de sus ideas filosóficas se encuentra el problema de la unión del entendimiento humano con Dios.
Su obra principal fue conocida en Occidente con el título de El filósofo autodidacto.
En ella, Abentofail estudia cómo es posible que el hombre en completa soledad pueda alcanzar la unión con Dio
Tras analizar las opiniones más importantes de los filósofos anteriores a él (Avempace, Algazel, Avicena, Alf
Prólogo del autor.
```

Figura 2-17. Ejemplo: Archivo de entrenamiento OpenNLP – Detección de oraciones
Elaboración: El autor de la tesis

El código completo de este ejemplo se encuentra en el anexo 4.

2.3.2. Tokenizador

Extrae una secuencia de caracteres, que suelen ser palabras, signos de puntuación, número, prefijos entre otros.

Se utilizará la misma frase que se empleó en el ejemplo de detección de oraciones, además de ocupar el mismo proyecto:

1. En la ruta del proyecto, crear una carpeta, copiar el modelo a ocupar para la tokenización y el archivo de entrenamiento (es-token.bin, token_train.txt).
2. El archivo de entrenamiento token_train.txt, posee frases que se encuentran en una sola línea separadas por espacios, o por la siguiente etiqueta <SPLIT>, algo similar a lo que se indica a continuación:

En el nombre de Dios<SPLIT>, clemente y misericordioso<SPLIT>! Bendiga Dios a nuestro Señor Mahoma y a su familia y compañeros<SPLIT>, y deles la paz<SPLIT>.

Has de saber<SPLIT>, pues<SPLIT>, que el que quiera alcanzar la verdad pura<SPLIT>, debe estudiar estos secretos y esforzarse por conocerlos<SPLIT>.

En el nombre de Dios<SPLIT>, clemente y misericordioso<SPLIT>! Bendiga Dios a nuestro Señor Mahoma y a su familia y compañeros<SPLIT>, y deles la paz<SPLIT>.

Filósofo y médico hispanoárabe<SPLIT>.

Figura 2-18. Ejemplo: Archivo de entrenamiento OpenNLP – Tokenización
Elaboración: El autor de la tesis

3. Se crea una clase, ésta puede tener cualquier nombre, en la misma se crea dos métodos uno para realizar el entrenamiento y el otro para realizar la tokenización.
4. El método de entrenamiento tiene las siguientes características:
    - Abrir un flujo de datos.

```
ObjectStream lineStream = new PlainTextByLineStream(new FileInputStream(RUTA_TEXTO), charset);
ObjectStream sampleStream = new TokenSampleStream(lineStream);

TokenizerModel model;
```

Figura 2-19. Codificación: Abrir flujo de datos OpenNLP – Tokenización
Elaboración: El autor de la tesis

    - Se llama al método de entrenamiento: TokenizerME.train

```
model = TokenizerME.train("es", sampleStream, true, TrainingParameters.defaultParams());
```

Figura 2-20. Codificación: Llamar método TokenizerME OpenNLP – Tokenización
Elaboración: El autor de la tesis

    - Guardar el modelo.

```
modelOut = new BufferedOutputStream(new FileOutputStream(MODELDIR));
model.serialize(modelOut);
```

Figura 2-21. Codificación: Guardar Modelo OpenNLP – Tokenización
Elaboración: El autor de la tesis

5. En el siguiente método se realiza la tokenización de la frase y contiene:
    - Primero cargar el modelo de tokenización, luego crear una instancia del modelo de TokenizaciónME

```
InputStream modelIn = new FileInputStream(MODELDIR);
try {

    TokenizerModel model = new TokenizerModel(modelIn);
    Tokenizer tokenizer = new TokenizerME(model);
```

Figura 2-22. Codificación: Instancia de TokenizaciónME OpenNLP – Tokenización
Elaboración: El autor de la tesis

- Se usa el método de “tokenize”, y se obtiene una matriz de cadenas como resultado.

```
String tokens[] = tokenizer.tokenize(TEXTO);
```

Figura 2-23. Codificación: Método tokenize OpenNLP – Tokenización
Elaboración: El autor de la tesis

- Sacar los tokens de la matriz

```
for (int i = 0; i < tokens.length; i++) {
    System.out.println("[" + (i + 1) + "]" + "==> " + tokens[i]);
}
```

Figura 2-24. Codificación: Presentación de tokens OpenNLP – Tokenización
Elaboración: El autor de la tesis

6. En la clase principal, se llama a los dos métodos que se creó previamente y los ejecutamos. El resultado del método que realiza la tokenización, sería lo que se indica en la figura 2-25:

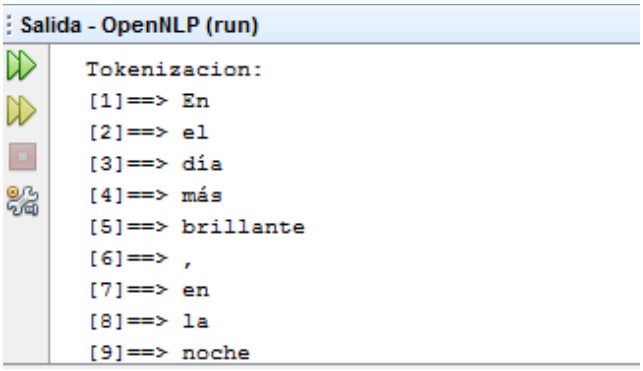

Figura 2-25. Resultado: Tokens OpenNLP – Tokenización
Elaboración: El autor de la tesis

En la figura 2-25, a la frase ya la ha divido en tokens, delimitados por un espacio o signos de puntuación.

Para una mayor ayuda sobre este módulo se consulta la documentación del API.

El ejemplo completo se encuentra en el anexo 5

2.3.3. POS tagging

El etiquetado gramatical, según la descripción del API, realiza el marcado de tokens con su correspondiente tipo de palabra basado en el propio token y contexto del mismo. El etiquetador de OpenNLP se basa en un modelo estadístico para lograr identificar cual sería la etiqueta.

Luego se indica el funcionamiento del API del etiquetador, se ocupará la misma frase para el análisis:

1. Crear una carpeta con cualquier nombre, en la ruta donde se crea el proyecto, en ella ubicar el modelo de aprendizaje y el de entrenamiento
2. El archivo de entrenamiento se representa en la figura 2-25:

El_DA Abogado_NC General_AQ del_SP Estado_NC ._Fc Daryl_VMI Williams_NC ._Fc subrayó_VMI hoy_RG la_DA necesidad_NC de_SP tomar_VMN medidas_NC para_SP proteger_VMN al_SP sistema_NC judicial_AQ australiano_AQ frente_RG a_SP una_DI página_NC de_SP internet_NC que_PR imposibilita_VMI el_DA cumplimiento_NC de_SP los_DA principios_NC básicos_AQ de_SP la_DA Ley_NC ._Fp
Melbourne_NP (_Fpa Australia_NP)_Fpt ._Fc 25_Z may_NC (_Fpa EFE_NC)_Fpt ._Fp
La_DA petición_NC del_SP Abogado_NC General_AQ tiene_VMI lugar_NC después_RG de_SP que_CS un_DI juez_NC del_SP Tribunal_NC Supremo_AQ del_SP estado_NC de_SP Victoria_NC (_Fpa Australia_NP)_Fpt se_P0 viera_VMS forzado_AQ a_SP disolver_VMN un_DI jurado_NC popular_AQ y_CC suspender_VMN el_DA proceso_NC ante_SP el_DA argumento_NC de_SP la_DA defensa_NC de_SP que_CS las_DA personas_NC que_PR lo_PP componían_VMI podían_VMI haber_VAN obtenido_VMP información_NC sobre_SP el_DA acusado_VMP a_SP través_NC de_SP la_DA página_NC CrimeNet_AQ ._Fp
Esta_DD página_NC web_AQ lleva_VMI un_DI mes_NC de_SP existencia_NC ._Fc tiempo_NC en_SP el_DA que_PR ha_VAI sido_VSP visitada_VMP en_SP más_RG de_SP un_DI millón_NC de_SP ocasiones_NC ._Fc y_CC facilita_VMI información_NC sobre_SP miles_PN de_SP crímenes_NC y_CC criminales_VMM ya_RG enjuiciados_VMP o_CC aún_RG perseguidos_VMM ._Fc datos_NC que_PR salen_VMI de_SP artículos_NC de_SP periódicos_NC y_CC archivos_NC judiciales_AQ ._Fp

Figura 2-26. Ejemplo: Archivo de entrenamiento OpenNLP – Tagging
Elaboración: El autor de la tesis

3. Se crea una clase con un nombre que la pueda identificar. Además, en esta clase se creará dos métodos; uno para el entrenamiento y otro para realizar el tagging.
4. El método de entrenamiento posee características similares al de entrenamiento de la tokenización, con pequeños cambios, es necesario seguir tres pasos para utilizarlo:
   - Primero abrir un flujo de datos

```
dataIn = new FileInputStream(RUTA_TEXTO);
ObjectStream<String> lineStream = new PlainTextByLineStream(dataIn, "UTF-8");
ObjectStream<POSSample> sampleStream = new WordTagSampleStream(lineStream);
```

Figura 2-27. Codificación: Abrir flujo de datos OpenNLP – Tagging
Elaboración: El autor de la tesis

   - Luego llamar al método de entrenamiento: POSTagger.train

```
model = POSTaggerME.train("es", sampleStream, TrainingParameters.defaultParams(), null, null);
```

Figura 2-28. Codificación: Método POSTagger OpenNLP – Tagging
Elaboración: El autor de la tesis

   - Se guarda el modelo.

```
modelOut = new BufferedOutputStream(new FileOutputStream(MODELDIR));
model.serialize(modelOut);
```

Figura 2-29. Codificación: Guardamos el modelo OpenNLP – Tagging
Elaboración: El autor de la tesis

5. Continuamos con el método que realizará el etiquetado del texto:

- Cargar el modelo en la memoria

```
InputStream modelIn = new FileInputStream(MODELDIR);

try {
    POSModel model = new POSModel(modelIn);
```

Figura 2-30. Codificación: Cargar modelo OpenNLP – Tagging
Elaboración: El autor de la tesis

- Crear instancias del modelo POSTaggerME

```
POSTaggerME tagger = new POSTaggerME(model);
```

Figura 2-31. Codificación: POSTaggerME OpenNLP – Tagging
Elaboración: El autor de la tesis

- Etiquetar cuando los datos se encuentren tokenizados.

```
ObjectStream<String> lineStream = new PlainTextByLineStream(new StringReader(TEXTO));
String line;

while ((line = lineStream.read()) != null) {

    String whitespaceTokenizerLine[] = WhitespaceTokenizer.INSTANCE.tokenize(line);
    String[] tags = tagger.tag(whitespaceTokenizerLine);
    for (int i = 0; i < tags.length; i++) {
        System.out.println("[" + (i + 1) + "]" + "==> " + "( " + tags[i] + " => " + whitespaceTokenizerLine[i] + " )");

    }

}
```

Figura 2-32. Codificación: Tokens etiquetados OpenNLP – Tagging
Elaboración: El autor de la tesis

6. Finalmente llamamos a estos dos métodos, a la clase que se creó previamente, y el resultado que se obtiene del segundo método, se indica en la figura 2-33:

```
run:
[1]==> ( SP => En )
[2]==> ( DA => el )
[3]==> ( NC => día )
[4]==> ( RG => más )
[5]==> ( AQ => brillante, )
[6]==> ( SP => en )
[7]==> ( DA => la )
[8]==> ( NC => noche )
```

Figura 2-33. Resultado: Texto etiquetado OpenNLP – Tagging
Elaboración: El autor de la tesis

Como ve en la figura 2-33, el texto ya se encuentra con su respectiva etiqueta, detalladas en el punto 2.7 de este capítulo.

El ejemplo completo se encuentra en el anexo 6.

2.3.4. Reconocerdor de la entidad (NER)

Este buscador tiene la funcionalidad de detectar entidades y números citados en el texto al igual que los mencionados anteriormente. También requiere un modelo dependiente al lenguaje y a la entidad en la que fue entrenado, se los puede encontrar es su página oficial para su uso posterior y para poder realizar esto se tiene que tener el texto tokenizado.

Puede ser necesario que se requiera entrenar el modelo con otros parámetros en el que se puede ocupar la siguiente etiqueta <START:person> Nombre <END> en el texto, y realizar cualquier modificación como se ve en la figura 2-33.

```
<START:Nombre> Pierre Vinken <END> , 61 años , se unirá a la junta como director no ejecutivo 29 de noviembre .
Sr . <START:Nombre> Vinken <END> es el presidente de Elsevier NV, el grupo editorial holandés .
No hay lugar más hermoso que la ciudad de <START:Lugar> Loja <END>
```

Figura 2-34. Ejemplo: Modelo de entrenamiento OpenNLP – NER
Elaboración: El autor de la tesis

Un ejemplo ayudará a una mayor comprensión de este módulo, se analizará la siguiente sentencia:

"Mario Correa, 23 años"

1. En el proyecto crear una nueva clase, y un método que realice este reconocimiento.
2. En la ruta donde se encuentra el proyecto, agregar una carpeta en la que constará el modelo del NER y el de entrenamiento si es necesario.
3. El método constará de las siguientes partes:
    - Cargar el modelo en memoria.

```
InputStream modelIn = new FileInputStream(MODELDIR);
try {

    TokenNameFinderModel model = new TokenNameFinderModel(modelIn);
```

Figura 2-35. Codificación: Cargar modelo – NER
Elaboración: El autor de la tesis

    - Crear instancias de NameFinderME

```
NameFinderME nameFinder = new NameFinderME(model);
```

Figura 2-36. Codificación: NameFinderME – NER
Elaboración: El autor de la tesis

    - Tokenizar el texto, y realizar la operación, detección de entidades

```
String whitespaceTokenizerLine[] = WhitespaceTokenizer.INSTANCE.tokenize(TEXTO);

Span nameSpans[] = nameFinder.find(whitespaceTokenizerLine);

for (Span s : nameSpans) {
    System.out.println(s.toString());
}
```

Figura 2-37. Codificación: Detección de entidades – NER
Elaboración: El autor de la tesis

4. Se hace el llamado al método para obtener los resultados

```
run:
[0..2) person
 [Mario Correa]
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figura 2-38. Resultado: Detección de entidades personas OpenNLP – NER
Elaboración: El autor de la tesis

Como se ve en la figura 2-38, ha localizado los que se pueden considerar como nombres comunes.

Al ejemplo completo está en el anexo 7.

2.3.5. Parser

Este análisis trata de agrupar las palabras de un texto a base de la similitud que poseen sus etiquetas.

Se indica un ejemplo del funcionamiento de este módulo, ocupando una frase más corta, que se utilizó en el apartado 2.2.3.

"El reino canta muy bien."

Seguir los siguientes pasos:

1. Crear una carpeta en la ruta o dirección donde se encuentra el proyecto, ubicar el modelo y el archivo de entrenamiento si es necesario.
2. Agregar una clase en la que conste un método para ejecutar este análisis o parser.
3. El método que realiza el análisis, tener en cuenta con lo siguiente:
    - Una instancia al modelo del analizador para cargarlo

```
InputStream modelIn = new FileInputStream(MODELDIR);
try {
    ParserModel model = new ParserModel(modelIn);
```

Figura 2-39. Codificación: Cargar modelo OpenNLP – Parsing
Elaboración: El autor de la tesis

- Crear una instancia del analizador a través del método ParserFactory

```
opennlp.tools.parser.Parser parser = ParserFactory.create(model);
```

Figura 2-40. Codificación: Método ParserFactory OpenNLP – Parsing
Elaboración: El autor de la tesis

- Agregar el texto al método que permite analizarlo.

```
Parse topParses[] = ParserTool.parseLine(TEXTO, (opennlp.tools.parser.Parser) parser, 1);
```

Figura 2-41. Codificación: Parser de la cadena OpenNLP – Parsing
Elaboración: El autor de la tesis

- Finalmente, el método show permite presentar el árbol.

```
for (Parse p : topParses){
    p.show();
}
```

Figura 2-42. Codificación: Presentación del árbol con el método show OpenNLP – Parsing
Elaboración: El autor de la tesis

4. Hacer el llamado de este método y presentar el análisis de la sentencia.

```
run:
(TOP (SENTENCE (SN (DA0MS0 El) (GRUP.NOM (NCMS000 reino))) (GRUP.VERB (VMIP3S0 canta)) (S (SADV (GRUP.ADV (RG muy)))) (PARTICIPI (CC bien))))

BUILD SUCCESSFUL (total time: 4 seconds)
```

Figura 2-43. Resultado: Árbol sintáctico de la sentencia OpenNLP – Parsing
Elaboración: El autor de la tesis

Se demuestra en la figura 2-43 el resultado del análisis, presentando al texto divido en grupos con su respectiva estructura gramatical.

Al ejemplo completo se encuentra en el anexo 8.

2.4. NLTK[11]

Natural Lenguaje Toollkit (NLTK), es sin duda una de las mejores plataformas para trabajar con textos en estado natural, por proporcionar una gran cantidad de recursos léxicos y corpus. Además ofrece herramientas como:

- Tokenización
- POS
- Taggging
- Razonamiento semántico

[11] http://www.nltk.org/

NLTK se puedo usar en Windows, Linux y Mac OS X, trabaja con Python, porque le permite realizar grandes cosas con el lenguaje natural.

Antes de comenzar con el procesamiento de lenguaje natural con python se tiene que instalar esta librería de la siguiente forma:

Abrir la consola de python y escribir lo siguiente:

import nltk

nltk.downlaod()

Se abre una ventana, que se indica en la figura 2-44

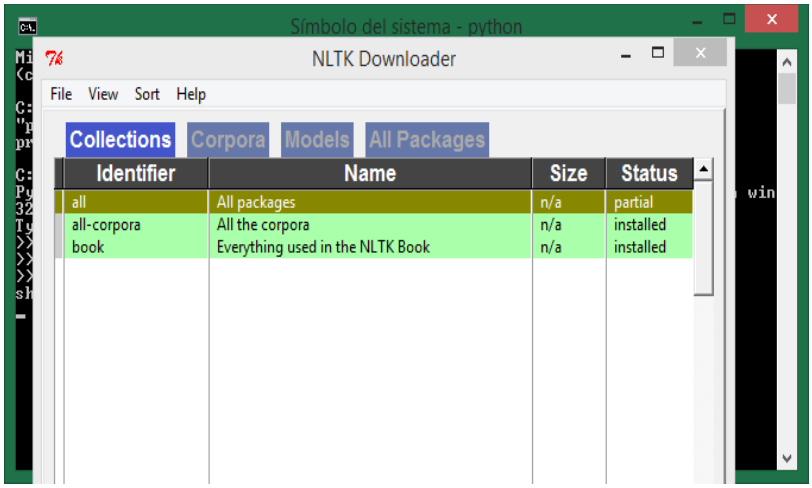

Figura 2-44. Instalación nltk python

Elaboración: El autor de la tesis

Se descarga e instala los paquetes necesarios para trabajar. Además no se necesita tener todos los corpus y modelos, se puede escoger y solo descargar los necesarios, como se ve en la figura 2-44. A continuación unos ejemplos con esta librería.

2.4.1. Tokenizador

Como se ha mencionado en las herramientas anteriores, es la separación en fragmentos más pequeños, determinados por un espacio o un signo de puntuación.

Se indica un ejemplo de tokenización, en la figura 2-45.

```
import nltk


def token(texto):
    text = nltk.word_tokenize(texto)
    for t in range(len(text)):
        print "Token_"+str(t+1)+": "+text[t]
    return text

if __name__ == "__main__":
    oracion = """This is an example"""
    token(oracion)
```

Figura 2-45. Tokenización nltk python

Elaboración: El autor de la tesis

Produce un resultado como el siguiente

```
Resultado:
Token_1: This
Token_2: is
Token_3: an
Token_4: example
```

:

Figura 2-46. Resultado tokenización nltk python

Elaboración: El autor de la tesis

En la figura 2-46, se aprecia la frase ya tokenizada.

2.4.2. POS Taggger

Permite procesar una secuencia de palabras y las categoriza según interprete a que grupo pertenece.

Adaptando el ejemplo anterior, se realiza el etiquetado de las palabras:

```
__author__ = "Mario"
__date__ = "$29/04/2015 04:55:30 PM$"

import nltk

def tagged(texto):
    text = nltk.word_tokenize(texto)
    tagged = nltk.pos_tag(text)

    return tagged

if __name__ == "__main__":
    oracion = """This is an example"""
    p = tagged(oracion)
    print p
```

Figura 2-47. Tagger nltk python

Elaboración: El autor de la tesis

Con el siguiente resultado:

```
[('This', 'DT'), ('is', 'VBZ'), ('an', 'DT'), ('example', 'NN')]
```

Figura 2-48. Resultado tagger nltk python

Elaboración: El autor de la tesis

2.4.3. NER

Puede reconocer sintagmas nominales y entidades sean éstos nombres, lugares, organizaciones u otras.

Siguiendo con el ejemplo anterior con pequeñas modificaciones, se reconoce a la entidad:

```
import nltk


def tagged(texto):
    text = nltk.word_tokenize(texto)
    tagged = nltk.pos_tag(text)

    return tagged

if __name__ == "__main__":
    oracion = """This is an example, I'm Mario"""
    p=tagged(oracion)
    entidades = nltk.chunk.ne_chunk(p)
    print entidades
```

Figura 2-49. Cunker-Ner nltk python

Elaboración: El autor de la tesis

El resultado que presenta este cambio es el siguiente:

```
Salida - NLTK
  (S
    This/DT
    is/VBZ
    an/DT
    example/NN
    ,/,
    I/PRP
    'm/VBP
    (PERSON Mario/NNP))
```

Figura 2-50. Resultado chunker-ner nltk python

Elaboración: El autor de la tesis

Se puede entrenar corpus para otros idiomas pero éste tiene un buen funcionamiento con el idioma inglés.

2.5. Freeling[12]

Herramienta que permite el análisis de texto, enfocado en los desarrolladores, es decir, que no es de fácil uso. Pero esto no quiere decir que un usuario con pocos conocimientos en programación, no logre obtener el análisis de un texto.

Está implementada para los siguientes idiomas, australiano (as), catalán (ca), inglés (en), francés (fr), gallego (gl), italiano (it), portugués (pt), ruso (ru), eslovaco (sl), español (es) y gales (cy). Y los servicios que ofrece para cada idioma están representados en la tabla 2-3:

Tabla 2-3. Servicios disponibles para los idiomas freeling

	as	ca	en	fr	gl	it	Pt	ru	sl	es	cy
Tokenización	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
División sintáctica	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Detección numérica		✓	✓		✓	✓	✓	✓		✓	
Detección de fecha		✓	✓		✓		✓	✓		✓	
Diccionario morfológico	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Reglas fijas	✓	✓	✓	✓	✓	✓	✓			✓	✓
Detección de multipalabra	✓	✓	✓	✓	✓	✓	✓			✓	✓
Detección de entidad básica	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
B-I-O detección de entidad básica		✓	✓		✓		✓			✓	
Detección del nombre de la entidad		✓	✓				✓			✓	
Detección de cantidad		✓	✓		✓		✓	✓		✓	
POS	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Codificación de fonética			✓							✓	
Sentido de la anotación		✓	✓	✓	✓				✓	✓	
Desambiguación del sentido		✓	✓	✓					✓	✓	
Análisis sintáctico superficial	✓	✓	✓		✓		✓			✓	
Análisis de dependencias completo	✓	✓	✓		✓					✓	
Resolución de correferencia		✓								✓	

Fuente: (Padró, 2013)

Algo fundamental, esta librería se encuentra desarrollada completamente en C++, y se puede compilar en otros sistemas operativos que no sean Unix.

[12] http://nlp.lsi.upc.edu/freeling/

También ofrece módulos para tratar el texto, explicaremos el funcionamiento de algunos, los cuales ayudaran en el desarrollo de este trabajo, que son los siguientes: la tokenización, la división, el POS tagger y el reconocedor de la entidad (ner).

2.5.1. Tokenizador

La documentación de este api dice: que este módulo ofrece la transformación de un texto plano en un objeto, de acuerdo a un conjunto de reglas preestablecidas (Padró, 2013). Las cuales se basan en tres secciones:

- Macros: Sección que permite definir las expresiones regulares.
- RegExps: En esta sección se definen las reglas de tokenización. Estas son las expresiones regulares y son aplicadas en el orden de la definición.
- Abbreviations: Se definen las abreviaturas más comunes, que no tienen que ser separadas del punto tenemos los siguientes ejemplos sr., etc., dr.

2.5.2. Detección de oraciones

En la documentación, este módulo recibe una lista de palabras hasta que se haya encontrado el fin de la oración, que pueden ser enviadas por el módulo de tokenización o por cualquier otro medio (Padró, 2013).

Este módulo posee cuatro secciones:

- General: Se encuentran las opciones generales del divisor.
- Markers: Muestra los grupos de marcadores que se pueden clasificar como marcadores.
- SentenceEnd: Crea una lista de caracteres que pueden ser considerados como el final de la oración.
- SentenceStart: Lista de caracteres que pueden aparecer al principio de una oración.

2.5.3. POS tagger

Existe dos módulos que realiza el etiquetado, la aplicación decidirá cuál de estos va a utilizar, el primero es el etiquetador hmm_tagger y el otro el relax_tagger.

El hmm_tagger es un módulo que permite añadir restricciones manuales y es más rápido que el otro modelo. En cambio el relax_tagger es un sistema híbrido que integra conocimientos estadísticos.

2.5.4. Reconocedor de la entidad (NER)

Detecta secuencias de palabras que se encuentran en mayúsculas, y que poseen algunas palabras funcionales.

Consta de las siguientes secciones:

- FunctionWords: Enlista la función de palabras en las que pueden estar los nombres propios.
- NE_Tag: Contiene la etiqueta que se asignara a las entidades reconocidas.
- Ignore: Encuentra una lista de etiquetas que no son considerados. como una entidad cuando están en mayúsculas en el medio de una frase.
- Names: Posee una lista de lemas que pueden ser nombres.
- Affixes: Tiene palabras que pueden ser prefijos de las entidades.
- RE_NounAdj, RE_Closed y RE_DateNumPunct: Permite modificar las expresiones regulares por defecto para las etiquetas del análisis.
- TitleLimit: Indica la longitud de una frase que está escrita en mayúsculas y la puede considerar como un título.

2.6. Gate[13]

Esta es una herramienta para el desarrollo y despliegue de componentes que procesan el lenguaje natural, destacando en el análisis de textos de cualquier tamaño. Además es un software de código abierto. Este framework cuenta con un conjunto de módulos llamados ANNIE (A Nearly-New Information Extraction System), y son los siguientes:

- Tokenizador
- División de oraciones
- Etiquetador gramatical
- Reconocedor de nombres
- Etiquetador de correferencia

[13] https://gate.ac.uk/ie/

2.7. Nomenclatura de textos en español

Las herramientas analizadas, ocupan la nomenclatura propuesta por el grupo EAGLES para la anotación morfosintáctica de los lexicones para los idiomas europeos. En la tabla 2-4 se indica las etiquetas que nos presenta el analizador para el lenguaje en español.

Tabla 2-4. Estructura de las etiquetas EAGLE.

Etiquetas			
Posición	**Atributo**	**Valor**	**Código**
Posición 1	Posición 2	Posición 3	Posición 4

Fuente: (The Eagles Lexicon Interest Group, 2011)

Se puede ver que en la primera posición, se ubica el orden y la posición en la que aparecen los atributos, en la siguiente cambia dependiendo de la categoría que posee el atributo, luego se ubica el valor que toma dicho atributo, para finalmente en la siguiente encontrar los códigos que se ha establecido para la representación de dicho atributo.

2.7.1. Adjetivo

Palabra cuya función propia es modificar al sustantivo con el que concuerda en género y número, lo puede modificar directamente o puede ser a través del verbo. (Real Academia Española, 2015)

Tabla 2-5. Estructura de la etiqueta de adjetivos.

Adjetivos			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Adjetivo	A
2	Tipo	Calificativo	Q
		Ordinal	O
3	Grado	Aumentativo	A
		Diminutivo	D
		Comparativo	C
		Superlativo	S
4	Género	Femenino	M
		Masculino	F
		Común	C
5	Número	Singular	S
		Plural	P
		Invariable	N

6	Función	-	0
		Participio	P

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.2. Adverbio

Es una palabra invariable cuya función propia es la de complementar a un verbo, o a un adjetivo, o a otro adverbio, además de poder incidir sobre grupos nominales, preposicionales o sobre toda la oración. (Real Academia Española, 2015)

Tabla 2-6. Estructura de la etiqueta de adverbios.

Adverbio			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Adverbio	R
2	Tipo	General	G
		Negativo	N

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.3. Determinante

Es una palabra que tiene como función introducir el nombre en la oración y precisar su extensión significativa, indicando cuales o cuantas de las entidades se refieren al hablante o si estas son conocidas por los interlocutores. (Real Academia Española, 2015)

Tabla 2-7. Estructura de la etiqueta de determinantes.

Determinantes			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Determinante	D
2	Tipo	Demostrativo	D
		Posesivo	P
		Interrogativo	T
		Exclamativo	E
		Indefinido	I
		Articulo	A
3	Persona	Primera	1
		Segunda	2
		Tercera	3
4	Género	Masculino	M

		Femenino	F
		Común	C
		Neutro	N
5	Número	Singular	S
		Plural	P
		Invariable	N
6	Poseedor	Singular	S
		Plural	P

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.4. Nombre

Palabra con género inherente que denota personas, animales o cosas y es capaz de funcionar como núcleo del sujeto, siendo equivalente al sustantivo. (Real Academia Española, 2015)

Tabla 2-8. Estructura de la etiqueta de nombres.

Nombres			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Nombre	N
2	Tipo	Común	C
		Propio	P
3	Género	Masculino	M
		Femenino	F
		Común	C
4	Número	Singular	S
		Plural	P
		Invariable	N
5	Clasificación Semántica	Persona	SP
		Lugar	G0
		Organización	O0
		Otros	V0
6	Grado	Aumentativo	A
		Diminutivo	D

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.5. Verbo

Denota un proceso, estado o una acción, siendo capaz de funcionar como núcleo del predicado y cuyas desinencias expresan modo, tiempo, número y personas. (Real Academia Española, 2015)

Tabla 2-9. Estructura de la etiqueta de verbos.

Verbos			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Verbo	V
2	Tipo	Principal	M
		Auxiliar	A
		Semiauxiliar	S
3	Modo	Indicativo	I
		Subjuntivo	S
		Imperativo	M
		Infinitivo	N
		Gerundio	G
		Participio	P
4	Tiempo	Presente	P
		Imperfecto	I
		Futuro	F
		Pasado	S
		Condicional	C
		-	0
5	Persona	Primera	1
		Segunda	2
		Tercera	3
6	Número	Singular	S
		Plural	P
7	Género	Masculino	M
		Femenino	F

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.6. Pronombre

Es una palabra que funciona de igual forma que un sustantivo, pero con la diferencia de que este carece de contenido léxico propio y cuyo referente lo determina su antecedente o una situación comunicativa. (Real Academia Española, 2015)

Tabla 2-10. Estructura de la etiqueta de pronombres.

Pronombres			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Pronombre	P
2	Tipo	Personal	P
		Demostrativo	D
		Posesivo	X
		Indefinido	I
		Interrogativo	T
		Relativo	R
		Exclamativo	E
3	Persona	Primera	1
		Segunda	2
		Tercera	3
4	Género	Masculino	M
		Femenino	F
		Común	C
		Neutro	N
5	Número	Singular	S
		Plural	P
		Impersonal Invariable	N
6	Caso	Nominativo	N
		Acusativo	A
		Dativo	D
		Oblicuo	O
7	Poseedor	Singular	S

		Plural	P
8	Politeness	Polite	P

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.7. Conjunción

Es una palabra invariable que involucra diversos tipos de oraciones subordinadas o que une vocablos o secuencias sintácticamente equivalentes. (Real Academia Española, 2015)

Tabla 2-11. Estructura de la etiqueta de adjetivos.

Conjunciones			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Conjunción	C
2	Tipo	Coordinada	C
		Subordinada	S

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.8. Interjección

Es una palabra invariable, con autonomía sintáctica, permite al orador expresar sentimientos o sensaciones, que induce a la acción al oyente. (Real Academia Española, 2015)

Tabla 2-12. Estructura de la etiqueta de interjecciones.

Interjecciones			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Interjección	I

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.9. Preposición

Es una palabra invariable y átona cuya función consiste en introducir un sustantivo o un grupo nominal con el que forma un complemento que depende sintácticamente de otro elemento del enunciado. (Real Academia Española, 2015)

Tabla 2-13. Estructura de la etiqueta de preposiciones.

Proposiciones			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Adposición	S
2	Tipo	Preposición	P
3	Forma	Simple	S

		Contraída	C
4	Género	Masculino	M
5	Número	Singular	S

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.10. Puntuación

Son un conjunto de puntos que sirven para puntear. (Real Academia Española, 2015)

Tabla 2-14. Estructura de la etiqueta de signos de puntuación.

Puntuación			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Puntuación	F

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.11. Números

Es una expresión de una cantidad con relación a su unidad. (Real Academia Española, 2015)

Tabla 2-15. Estructura de la etiqueta de numerales.

Numerales			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Cifra	Z
2	Tipo	Partitivo	d
		Moneda	m
		Porcentaje	p
		Unidad	u

Fuente: (The Eagles Lexicon Interest Group, 2011)

2.7.12. Fecha y hora

Tabla 2-16. Estructura de la etiqueta de fecha y hora.

Fecha y Hora			
Posición	**Atributo**	**Valor**	**Código**
1	Categoría	Fecha/Hora	W

Fuente: (The Eagles Lexicon Interest Group, 2011)

Revisar el anexo 9, para tener una visión clara sobre las mismas.

2.8. Conclusiones

En la tabla 2-17, están representados los módulos por una sigla, tokenización (T), detección de oraciones (SD), etiquetador o POS tagging (POS), y reconocedor de la entidad (NER). Solo se describen los que tienen en común estas herramientas, además resume las herramientas analizadas para concluir lo siguiente:

En este capítulo se observa que cada una de estas herramientas tienen sus puntos fuertes pero en este caso se seleccionó OpenNLP que permite tratar los textos en español y además se encuentra desarrollado en Java lo cual facilita el desarrollo, Java es independiente de la plataforma en la que se está trabajando, además el aprendizaje es menos costoso, si se lo compara con otros lenguajes. OpenNLP también brinda una vasta documentación, además consume una menor cantidad de recursos y realiza el procesamiento en menor tiempo.

Tabla 2-17. Herramientas de extracción de información

	Modularidad	Interfaz Gráfica	Licencia	Idioma	Lenguaje de Programación	Complejidad	Módulos	Documentación Clara	Versión
Apache OpenNLP	Si	No	GNU	Multidioma	Java	Media	T, SD, POS, NER	Si	1.5.3
StanfordCoreNLP	Si	No	GNU	Multidioma	Java, Python, C++, otros	Media	T, SD, POS, NER	Si	3.5.2
Freeling	Si	No	GNU	Multidioma	C++, Python	Media	T, SD, POS, NER	Si	3.1
NLTK	Si	No	GNU	Ingles	Python	Media	T, SD, POS, NER	Si	3.0
GATE	Si	Si	GNU	Multidioma	Java	Media	T, SD, POS, NER	No	8.0

Elaboración: El autor de la tesis.

CAPÍTULO III

ADAPTACIÓN DE LA HERRAMIENTA AL CONTEXTO DE ESTUDIO

3.1. Introducción

Este capítulo se indica la adaptación y funcionamiento de la herramienta que servirá para el procesamiento del lenguaje y el etiquetado de sentimientos. Tomando en cuenta las herramientas que se analizó en el capítulo anterior y permitirán realizar pruebas.

3.2. Análisis del texto

Para poder realizar un análisis del texto se debe tomar en cuenta algunos factores que ayudan a determinar su correcta estructura, como los siguientes puntos:

- **Alteraciones lingüísticas.** La mayor parte de los documentos o mensajes que se encuentran en internet están llenos de errores ortográficos y tipográficos, usos indebidos de las reglas gramaticales, puntuación, acrónimos entre otros.
- **Exceso de signos de puntuación.** Es muy frecuente encontrar en mensajes o chats, signos como: comas (,), puntos (.), exclamaciones (¡!), interrogaciones (¿?), puntos suspensivos (…), que se ocupan sin ninguna restricción, sin regirse a ninguna regla gramatical.
- **Acrónimos.** Siendo este el que se forma utilizando la inicial de la palabras, en los mensajes se pueden encontrar acrónimos para acortar la misma oración.

3.3. Construcción del demo

Para esta adaptación fue necesario la construcción de un módulo que permita: la carga de datos, eliminar los “stopwords” y etiquetar los sentimientos.

Se usaron diccionarios de tipo raíz (Anexo 10), por la gran cantidad de conjugaciones que pueden salir de una palabra. Los cuales fueron elaborados, por sinónimos de la emoción principal.

Estos diccionarios se crearon con ayuda de bases de datos léxicas como Wordnet, EuroWordnet o WordReference[14]. Este último es un diccionario de sinónimos y antónimos.

En la construcción de este demo se tomó en cuenta el ingreso de datos, que puede ser masivo o en pocas cantidades. En la figura 3-1 se presenta un modelo general de la forma en la que se trabaja.

[14] http://www.wordreference.com/sinonimos/

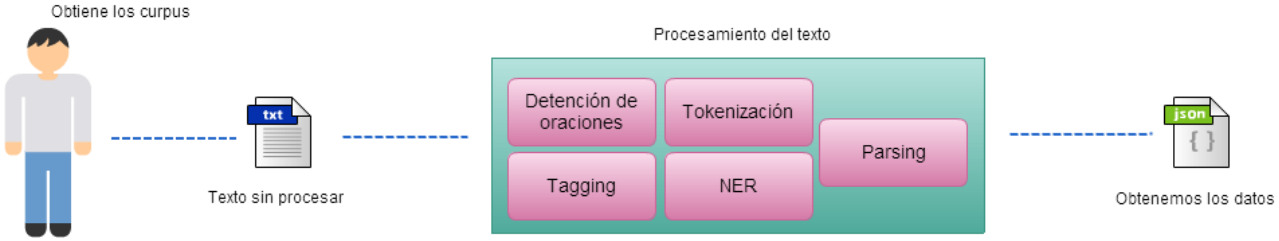

Figura 3-1. Forma de trabajo de la herramienta
Elaboración: El autor de la tesis.

Como se mencionó en el capítulo anterior se utilizó la biblioteca de OpenNLP, para el análisis del texto, que se compone de una gran variedad de módulos, que los representa en la figura 3-2.

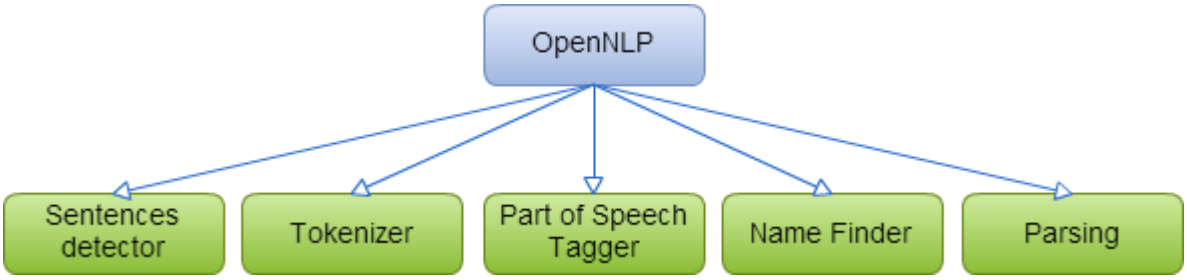

Figura 3-2: Módulos OpenNLP a ocupar
Elaboración: El autor de la tesis.

Se ha explicado el funcionamiento de estos módulos, ahora se hará una pequeña mención de lo que estos realizan:

- **Sentences detector:** permite encontrar las oraciones a partir de un delimitador que es el punto que indica el fin de la oración.
- **Tokenizer:** realizar la separación o segmentación del texto por medio de delimitadores.
- **Part of speech tagger:** se encarga de realizar el etiquetado de los tokens.
- **Name finder:** detecta nombres de personas u organizaciones. Dependiendo del lenguaje en el que se trabaja.
- **Parsing:** realiza la división del texto en partes que se encuentran sintácticamente relacionadas, por ejemplo en grupos de verbos.

Ahora se visualizará los requerimientos y casos de uso, para el desarrollo del proyecto.

3.3.1. Requerimientos

Los requerimientos identificados para el desarrollo de este proyecto son los siguientes:

- Ingresar texto o corpus.
- Presentar los datos.
- Realizar el etiquetado de sentimientos

- Descargar los resultados.

3.3.2. Casos de uso

En la figura 3-3, presentamos el esquema de los casos de uso para este proyecto. Basados en los requerimientos que se presentaron previamente.

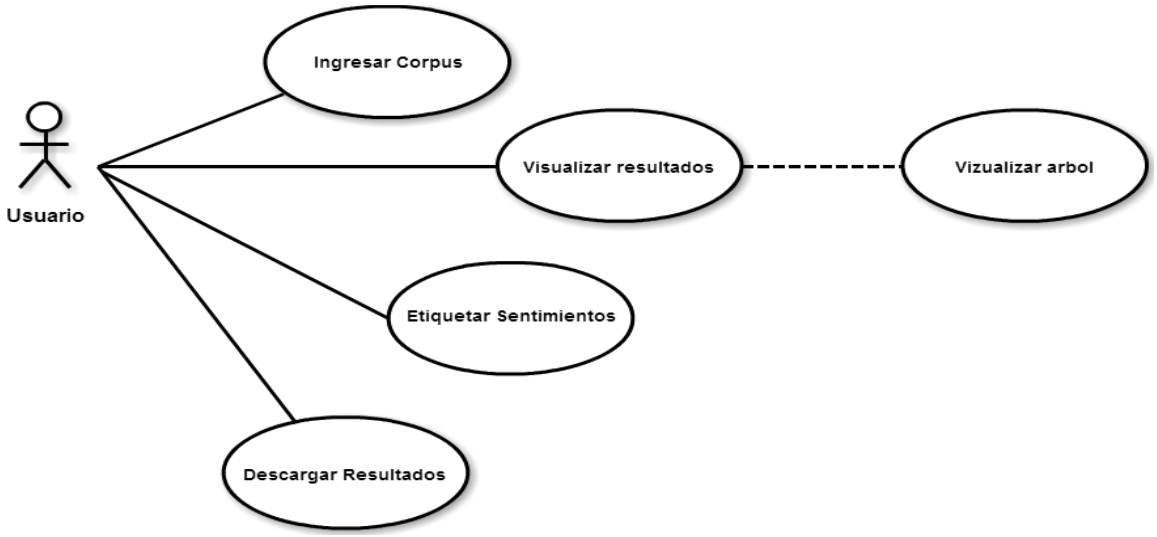

Figura 3-3: Diagrama general de casos de uso.
Elaboración: El autor de la tesis.

3.4. Resultado del aplicativo

3.4.1. Ingreso de corpus

El usuario ingresa el corpus o texto que desee analizar. Se puede hacer de dos formas:

a) Ingresando el texto directamente en la caja de texto
b) Cargar un archivo .txt

Estas dos opciones las podemos ver en la figura 3-4.

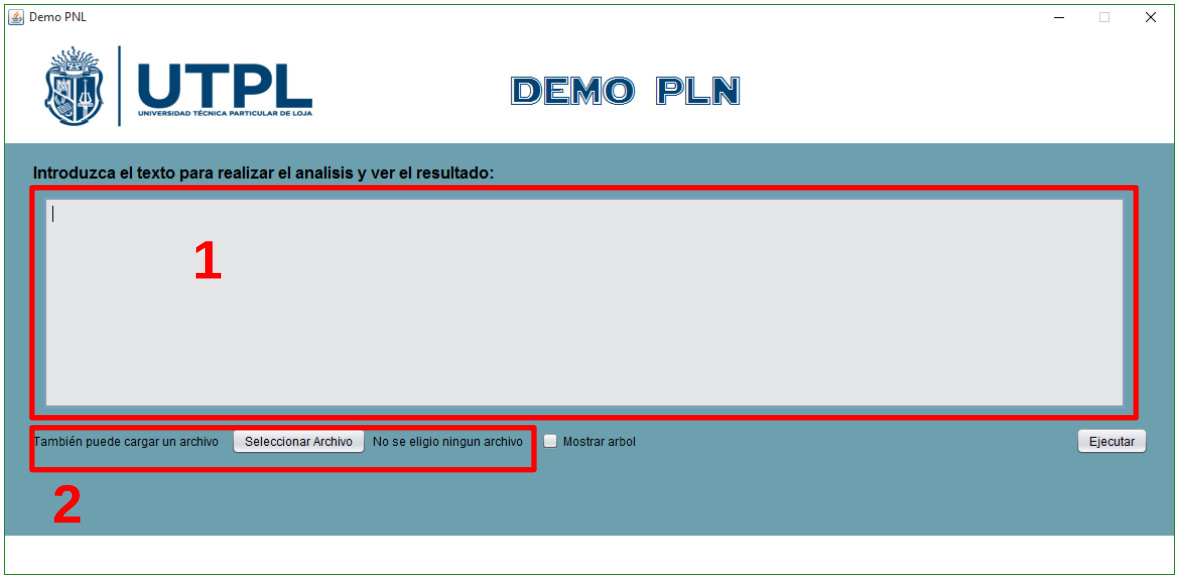

Figura 3-4. Opciones de carga de datos.
Elaboración: El autor de la tesis.

En la figura 3-4 se puede ver fácilmente donde se encuentran las dos formas de carga de información.

Asimismo, en la figura anterior se muestra la pantalla principal de la aplicación, en la que se ingresa el texto, y se realiza la acción de ejecutar. Previamente se puede seleccionar la opción de "Mostrar árbol", que permite realizar una análisis más completo de la frase y presentarlo como un árbol binario.

Se tomó como ejemplo cuatro mensajes encontrados en el la red social del EVA, en la materia de dibujo artístico.

Al ejecutarse, lo primero que se realiza, es separar por oraciones el contenido ingresado, como se indica en la figura 3-5.

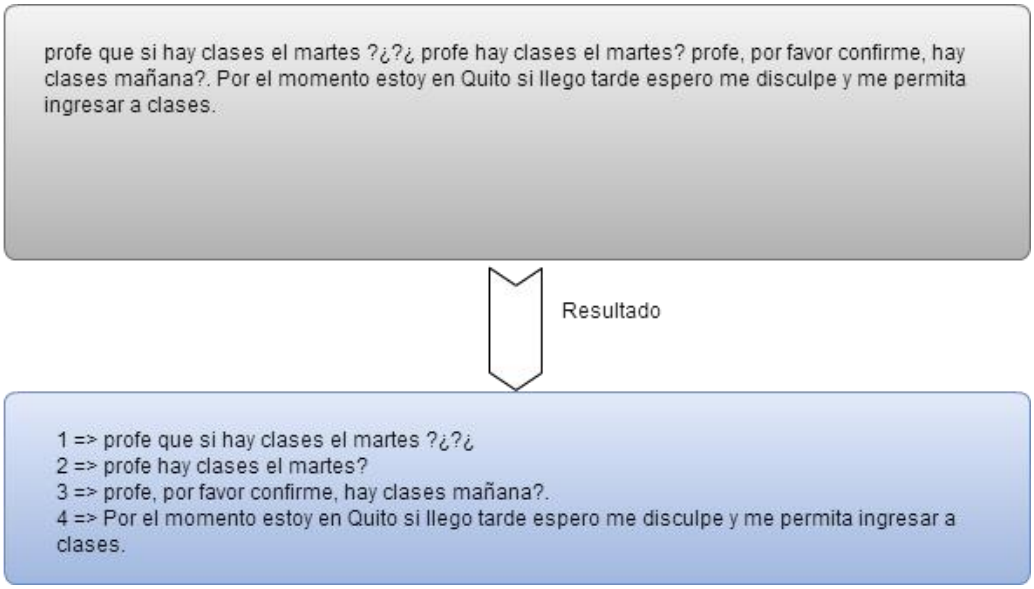

Figura 3-5. Detección de oraciones
Elaboración: El autor de la tesis.

Ahora con los datos del corpus separado por oraciones se procede a realizar el resto de las operaciones.

3.4.2. Presentación de datos

En una pantalla presentan un conjunto de botones, que son la cantidad de oraciones extraídas del corpus ingresado, como se puede ver en la figura 3-6.

Figura 3-6. Pantalla resultado de oraciones.
Elaboración: El autor de la tesis.

En la figura 3-6, se encuentra el corpus separado por oraciones, para ver el análisis de una oración se hace clic en la que se desea. Aparece una ventana en la que muestra un análisis más amplio, se aprecia en la figura 3-7.

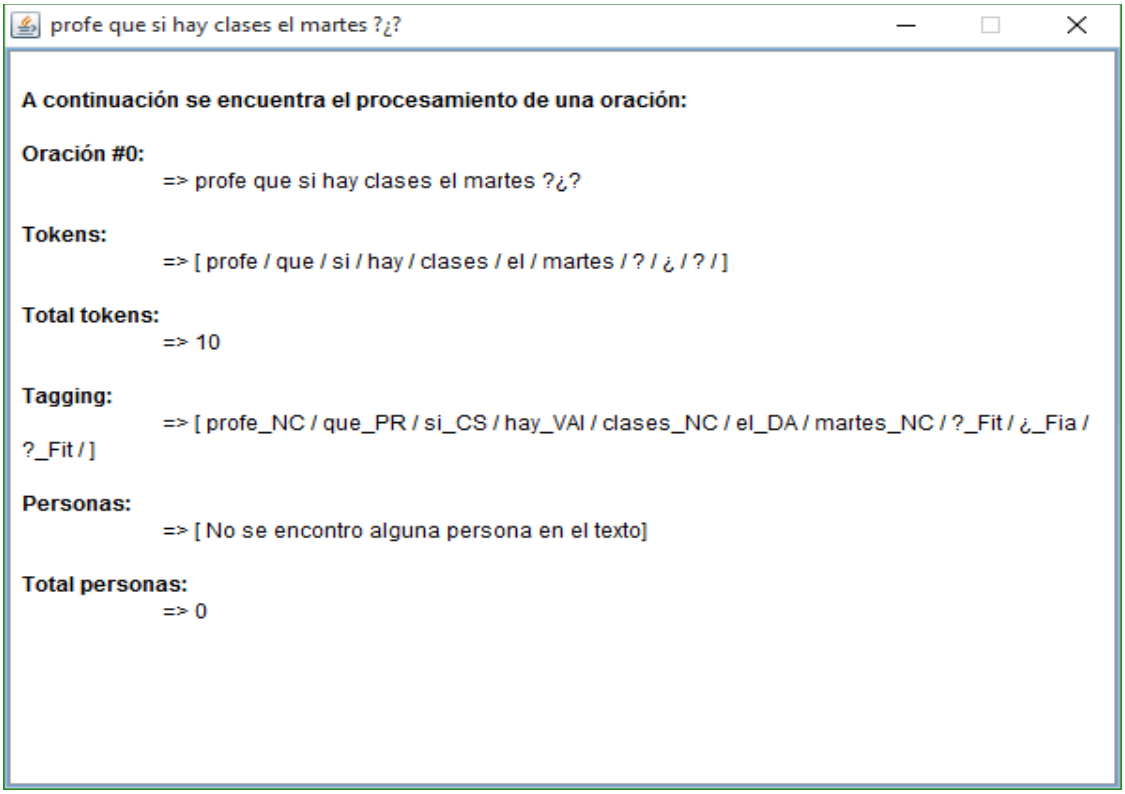

Figura 3-7. Análisis de la oración sin árbol
Elaboración: El autor de la tesis.

Como se ve en la figura 3-8, la presentación cambia si se marcó la opción de “mostrar árbol”. En la ventana aparece una nueva opción que permite presentar el resultado en forma de árbol, al estilo de un índice de contenidos.

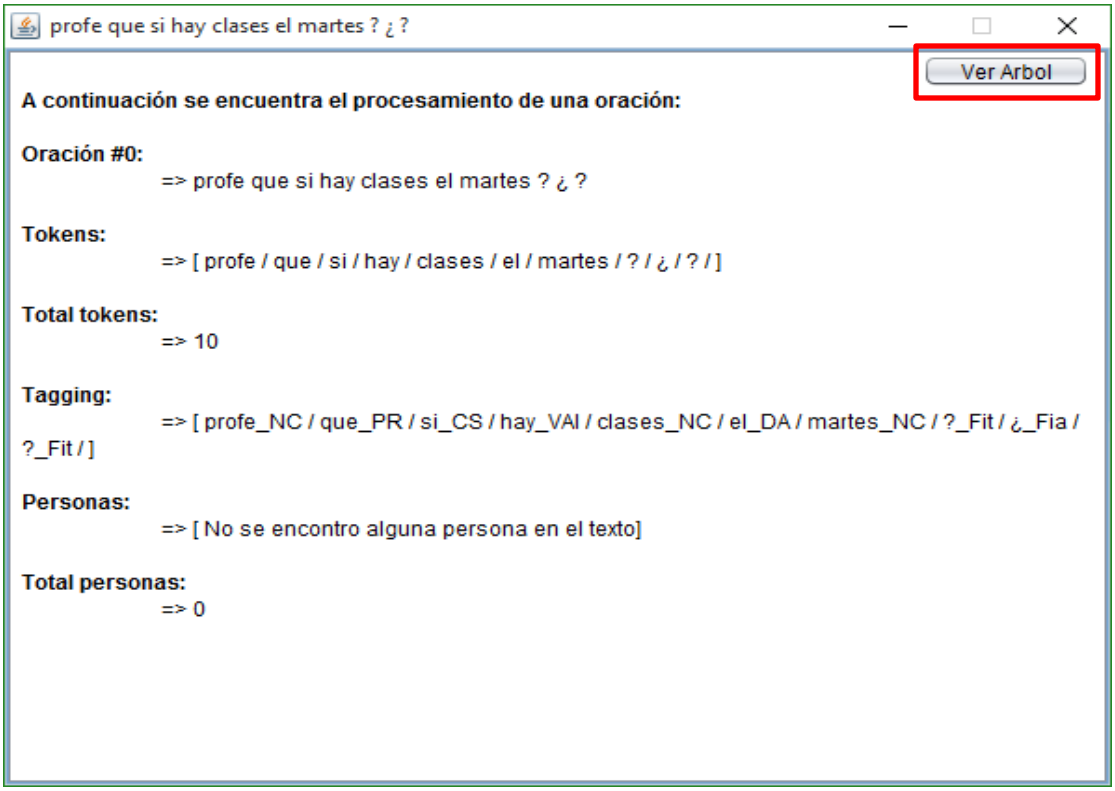

Figura 3-8. Análisis de la oración con árbol
Elaboración: El autor de la tesis.

La nueva ventana, se la ve en la figura 3-9.

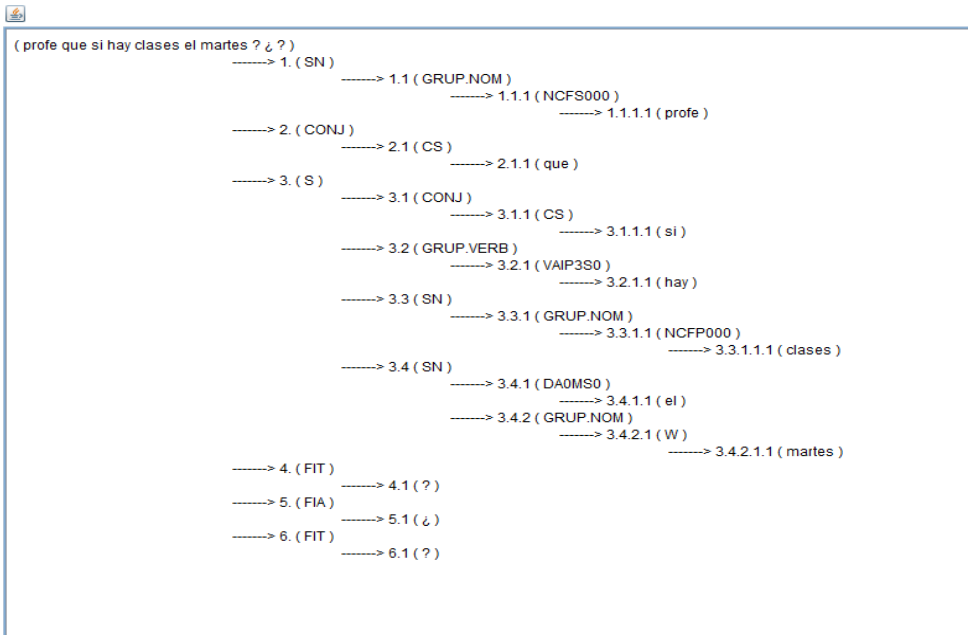

Figura 3-9. Estructura de árbol binario
Elaboración: El autor de la tesis.

En la figura 3-9, se observa la estructura de padre e hijo que posee la oración, a este árbol se lo obtuvo con ayuda de módulo de OpenNLP del parser. En la primera fila encontramos algunos

grupos como (SN) y (FIT), los cuales pueden tener como hijo un valor final o simplemente otro grupo, con ayuda de la tabla que tenemos en el anexo 9, en el caso de SN que es el sintagma nominal de la oración encontramos que posee un hijo que es un grupo de nombres y este también tiene un hijo que pertenece a la etiqueta de nombres comunes según el etiquetado EAGLE, la hoja final de esa rama es el valor que está representando la etiqueta (NCFS000), que en este caso es "profe". En cambio (FIT) es el código de la etiqueta para cerrar el signo de interrogación.

Regresando a las figuras 3-7 y 3-8, podemos visualizar que presentan los tokens que posee la oración, la cantidad de estos que existen en la misma, y el etiquetado que posee cada uno. Además, presenta el total de personas que existen en la oración.

3.4.3. Etiquetado de sentimientos

En la parte inferior de la figura 3-6, se encuentra la opción que realiza el etiquetado de sentimientos. Es necesario eliminar las palabras vacías (stopwords) del corpus ingresado con la finalidad de retirar las palabras que no aportan nada.

El listado de stopwords se encuentra en el anexo 11.

A las palabras restantes se las asocia con una de las siguientes emociones:

- Aburrimiento
- Angustia
- Ansiedad
- Confusión
- Frustración
- Simpatía

Este etiquetado, se lo hace con diccionarios de sinónimos, el resultado se encuentra en la figura 3-9.

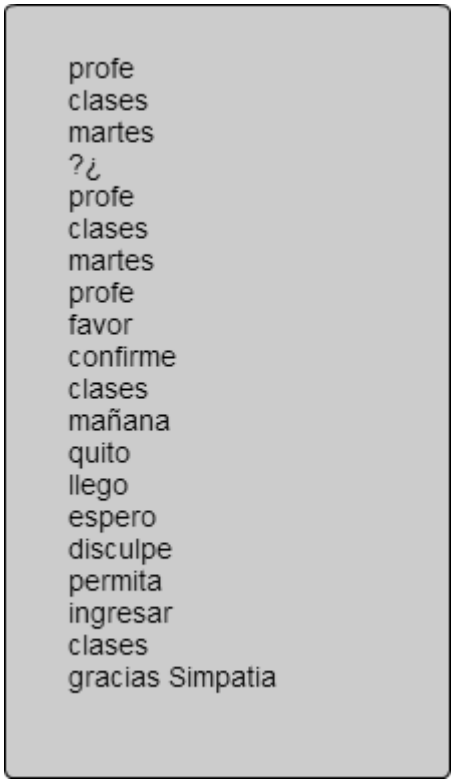

Figura 3-10. Etiquetado de sentimientos
Elaboración: El autor de la tesis.

El etiquetado crea un archivo .txt o .arff según la elección del usuario. Los datos que se guardan en el archivo son los que se encontró en los diccionarios ocupados para realizar esta operación.

En la figura 3-10, podemos encontrar un ejemplo de los datos que se guardan en el archivo.

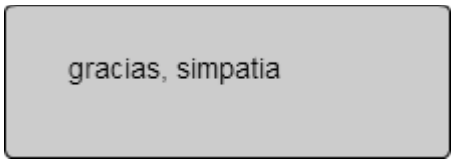

Figura 3-11. Datos de archivo etiquetado
Elaboración: El autor de la tesis.

3.4.4. Descarga de la información

En la figura 3-6, que corresponde a la pantalla de resultados de las oraciones, se puede ver en la parte inferior un botón con la descripción de “Descargar”, nos permite sacar una copia de la información en un archivo .json como se ve en la figura 3-12.

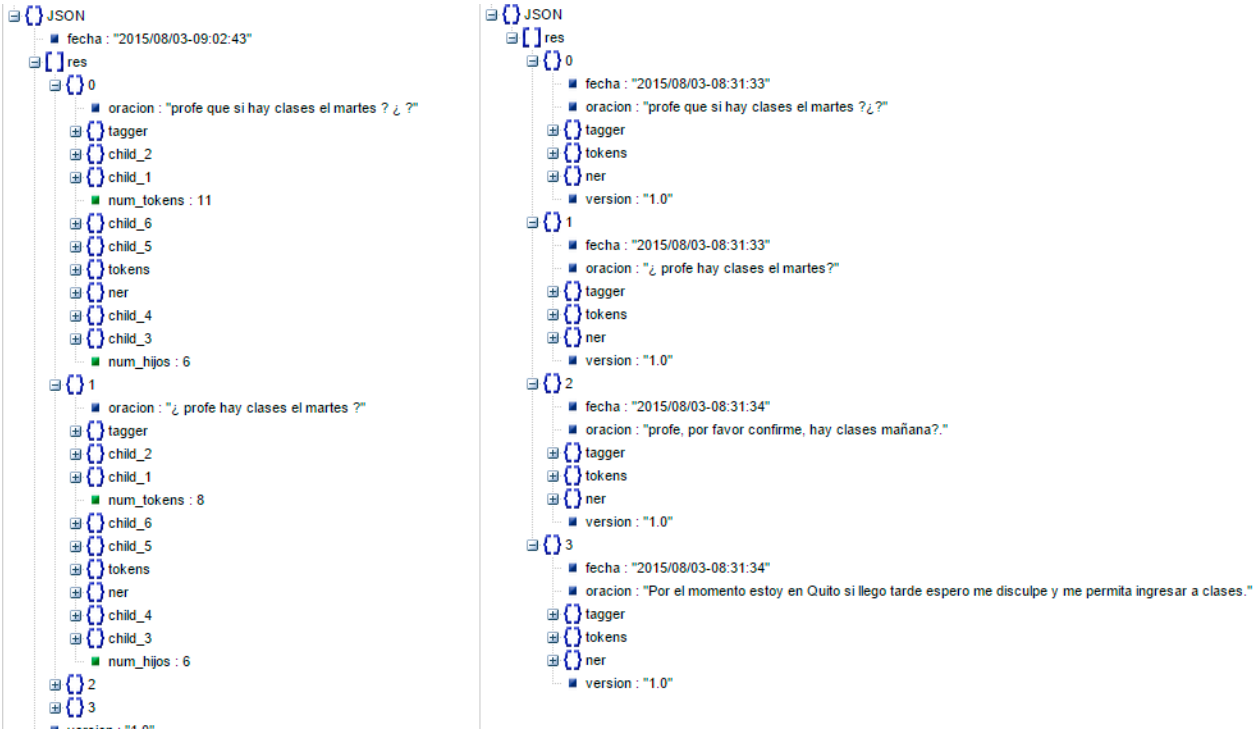

Figura 3-12. Archivo Json con los resultados
Elaboración: El autor de la tesis.

CAPÍTULO IV

EXPERIMENTACIÓN Y ANÁLISIS DE RESULTADOS DEL FUNCIONAMIENTO DE LA HERRAMIENTA

4.1. Introducción

En el presente capítulo se mostrara el funcionamiento de la aplicación, con un conjunto de datos más amplio. Para obtener nuevos corpus etiquetados.

4.2. Comprobación de modelos OpenNPL

Se comprobó la precisión de los modelos, detección de oraciones, tokenización, etiquetado y reconocedor de la entidad. Estos resultados se detallan en la tabla 4.1. Los dos primeros se obtuvieron mediante el API de evaluación que tiene OpenNPL ocupando las siguientes líneas de comando respectivamente:

- opennlp-tools-1.5.3.jar SentenceDetectorEvaluator -model es-sent.bin -data es-sent.eval -encoding UTF-8
- opennlp-tools-1.5.3.jar TokenNameFinderEvaluator -model es-token.bin -data es-token.eval -encoding UTF-8

Los archivos es-sent.eval y es-token.eval poseen la misma información que los archivos ocupados para entrenar estos modelos, detallados en la sección 2.3.1 y 2.3.2 respectivamente.

Para el etiquetado, luego de haber realizado algunas pruebas con algunos modelos, se decidió ocupar el desarrollado por (Caicedo Carvajal, 2012), el presenta el resultado de la evaluación de este modelo.

Finalmente, para el reconocedor de la entidad revisamos el plan de pruebas[15] que realizó Apache OpenNLP, para el entrenamiento de este modelo.

Tabla 4-1. Comprobación de precisión de modelos

Criterios	Precisión	Porcentaje
Detección de oraciones	0,9743538	97%
Tokenización	0,9989394	99%
Etiquetado	0,9629507	96%
Ner	0,9195205	92%

Elaboración: El autor de la tesis.

Se realizó la validación de los resultados con un conjunto de datos que ya se encuentran etiquetados morfológicamente. Con ayuda de la herramienta de Freeling, se han formado cuatro corpus para realizar esta experimentación, los cuales se detallan en la tabla 4-2.

[15] https://cwiki.apache.org/confluence/display/OPENNLP/TestPlan1.5.3

Tabla 4-2. Textos de prueba

Corpus	Total de oraciones	Total de tokens	Total de etiquetas
Datos_1	4	168	168
Datos_2	4	123	123
Datos_3	4	174	174
Datos_4	4	128	128

Elaboración: El autor de la tesis.

La estructura que tienen los corpus descritos en la tabla 4.2, es la siguiente: Constan de cuatro oraciones de temas variados, el etiquetado gramatical de las oraciones, los tokens que tienen y el total global de los mismos.

Por ejemplo, se tomó una fracción de una oración del corpus:

- "Tampoco descartaron que la suspensión de las negociaciones…"

Luego, tiene el etiquetado de la oración, siguiendo con el ejemplo:

- Tampoco_**RG** descartaron_**VMIS3P0** que_**CS** la_**DA0FS0** suspensión_**NCFS000** de_**SPS00** las_**DA0FP0** negociaciones_**NCFP000**…

Después, la cantidad de tokens que la oración posee, en este ejemplo seria:

- **Total tokens:** 37

Finalmente, la cantidad de tokens que tiene el corpus, obtenida sumando el total de tokens que tienen las oraciones que conforman el corpus, en este caso sería:

- **Total tokens del corpus:** 168

Estos datos se encuentran en el anexo 11.

Se realiza una comparación con los datos que tenemos en el anexo 11, para ver cómo está el funcionamiento de nuestros módulos.

Tabla 4-3. Análisis morfológico

Módulos	OpenNLP	
	Aciertos	Fallos
Detección de oraciones	100%	0%
Tokenizador	99,36%	0,64%
Etiquetador	96,25%	3,75%

Elaboración: El autor de la tesis.

Con estos resultados se concluye que, los modelos están trabajando de una forma correcta con referencia a los datos obtenidos con la herramienta Freeling.

4.3. Resultados con datos reales

En esta prueba, se ocupó dos bancos de mensajes de la red social del EVA de la modalidad abierta, del primer banco de mensajes se logró obtener 152 oraciones, un total de 5543 tokens, y 23 se encuentran catalogados como personas. Del segundo archivo se obtuvo 185 oraciones y 4237 tokens, encontrando en el mismo 11 que están catalogados como personas. A partir de estos tokens se procedió a crear un nuevo corpus con un etiquetado de sentimientos. Obteniendo un archivo .arff para ser ejecutado en weka o simplemente un archivo .txt.

En la figura 4.1 podemos ver estos resultados de forma gráfica.

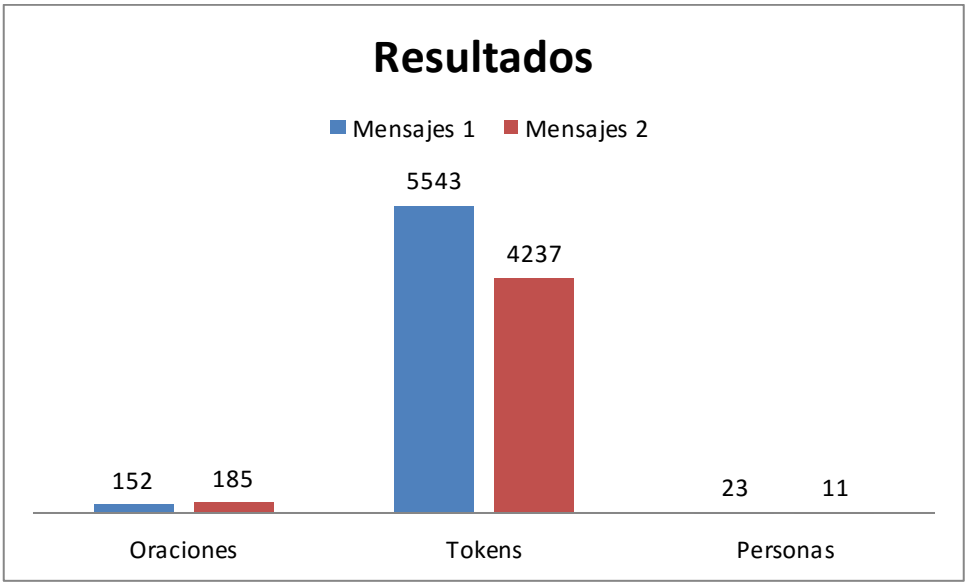

Figura 4-1. Gráfico de resultados generales del análisis
Elaboración: El autor de la tesis.

A las etiquetas no se las consideró para este gráfico porque numéricamente vendrían a ser igual a la cantidad de tokens encontrados.

Al analizar una oración, de cualquiera de los dos bancos, tiene una etiqueta asignada a cada palabra o mejor dicho cada token. En la tabla 4-1 se visualiza este ejemplo, la descripción la encontramos en el anexo 10, que corresponde a las etiquetas Eagle.

Tabla 4-4. Etiquetado de una oración

"No pude presentar el trabajo final debido a que estaba de turno."		
Token	**Etiqueta**	**Descripción**
No	RN	Adverbio negativo
puede	VMIS1S0	Verbo principal indicativo
presentar	VMN0000	Verbo principal infinitivo
El	DA0MST0	Artículo determinante
trabajo	NCMS000	Nombre común
final	AQ0CS0	Adjetivo calificativo
debido	VMP00SM	Adjetivo calificativo
a	SPS00	Preposición
que	CS	Pronombre relativo
estaba	VMII3S0	Verbo principal infinitivo
De	SPS00	Preposición
turno	NCMS000	Nombre común
.	FP	Punto

Elaboración: El autor de la tesis.

Este resultado también se lo presenta en forma de árbol binario, con la estructura de un índice de libro, como se ve en la figura 4-2.

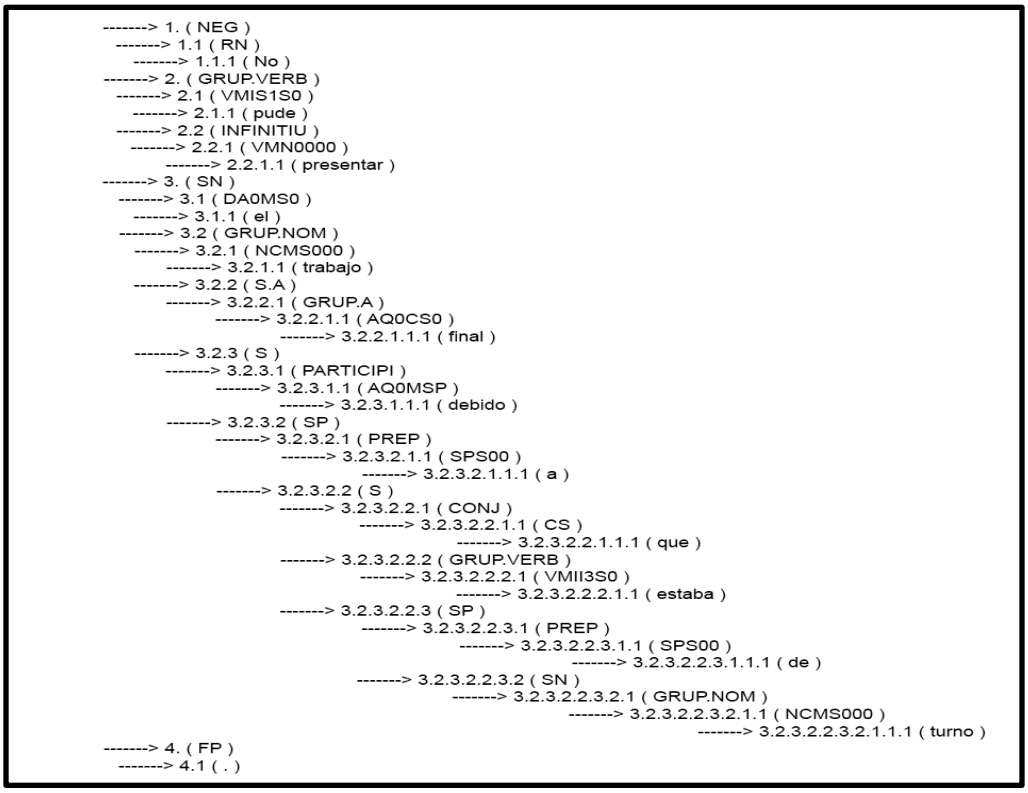

Figura 4-2. Árbol binario
Elaboración: El autor de la tesis.

En la figura 4-2, aparte de indicar las etiquetas gramaticales, también lo ha divido la oración en grupos que representan de forma más apropiada a las partes de la oración.

El etiquetado de sentimientos de los bancos de mensajes, lo encontramos resumido en la tabla 4-5.

Tabla 4-5. Resumen del etiquetado de sentimientos

Sentimientos encontrados en los mensajes	
Emoción	**Veces encontrado**
Aburrimiento	6
Angustia	105
Ansiedad	7
Confusión	6
Frustración	19
Simpatía	202

Elaboración: El autor de la tesis.

En la tabla 4-5, se ve la cantidad de veces que una emoción se ha encontrado en el texto, indicando que las más altas son: Simpatía y Angustia respectivamente, revelando que los estudiantes tienen un alto nivel de agradecimiento, pero igual necesitan de una retroalimentación de los contendidos.

Las pruebas realizadas en este capítulo son para verificar el funcionamiento de la aplicación, se pueden hacer otras para verificar la calidad de resultados que devuelve la aplicación, también para ver el tiempo de respuesta que puede tener con una gran cantidad de datos.

CONCLUSIONES

Al terminar el presente trabajo se puede concluir lo siguiente:

- Se implementó satisfactoriamente una herramienta para el procesamiento del lenguaje natural en el idioma español. En esta herramienta es necesario ingresar el texto con una estructura correcta, de lo contrario el análisis no será válido.
- En el capítulo II se analizaron algunas herramientas para PLN, que estaban desarrollas en varios lenguajes. Sin embargo se utilizó APIs desarrolladas en Java debido a que este lenguaje ofrece una gran variedad de ventajas como: Independencia de plataforma, alto rendimiento y facilidad de aprendizaje.
- Las herramientas analizadas en el capítulo II, tienen una licencia open source (licencia GNU), que ofrecen en algunos casos mayores beneficios que los de uso comercial, ya que poseen un desarrollo por parte de la comunidad, pero también pueden tener una mala documentación o una solo básica. Por eso se eligió OpenNLP, porque posee una amplia documentación y consume una menor cantidad de recursos (tiempo y hardware).
- Para el etiquetado de sentimientos, es conveniente ocupar StopWords, porque permiten eliminar palabras irrelevantes que se encuentran en el texto, ya que no representan ninguna emoción.
- Al trabajar con distintas herramientas existe el riesgo de que éstas no se adapten, produciendo inestabilidad en el sistema final, llegando a tener tiempos de respuesta más elevados y generar mayor cantidad de inconvenientes a la hora de depurar.
- Se pueden encontrar varios recursos lingüísticos como WordNet o EuroWordNet que han sido tratados en otras investigaciones, las que se podría aprovechar para poder obtener mejores resultados, además facilitan la obtención de sinónimos, que ayudan en la creación de diccionarios, fueron utilizados en este trabajo para el etiquetado de sentimientos.
- Los resultados de precisión de los modelos: detección de oraciones, tokenización, etiquetado y ner, están en un rango del 92% al 99%, al aplicar las pruebas, se obtuvo resultados favorables con referencia a la herramienta freeling.
- Las emociones que se etiquetaron con mayor frecuencia en el texto analizado son: Simpatía y Angustia. Estas emociones indican que los estudiantes agradecen las indicaciones pero a su vez necesitan una retroalimentación de los conocimientos.
- La aplicación desarrollada, puede realizar el PLN de una gran cantidad de texto o corpus, pero mientras mayor sea la cantidad, el tiempo de análisis también será mayor.

RECOMENDACIONES

- Al trabajar con herramientas de licencia GNU, se debe revisar que éstas tengan la documentación necesaria, como manuales descriptivos, libros. Además de tener asistencia vía correos y de ser posible que posea una wiki.
- Es importante al momento de seleccionar una herramienta de procesamiento de lenguaje natural, verificar los idiomas a los que brinda soporte.
- No ocupar demasiadas herramientas de distintos proveedores porque pueden presentar dificultades al momento de realizar la integración de las mismas, pueden consumir más recursos de los necesarios.
- Para trabajar con proyectos basados en el análisis de texto en diferentes idiomas, se debe hacer con ayuda de lingüistas o expertos en temas que tengan que ver con análisis de la gramática.

TRABAJOS FUTUROS

A esta adaptación se pueden generar nuevos proyectos, los cuales son desarrollados en el futuro. El procesamiento de lenguaje natural nos ofrece un gran campo:

- Realizar una adaptación de la herramienta para un entorno web, para darle una mejor presentación a los resultados e intentando que se un poco más amigable para el usuario.
- Realizar la implementación de análisis de sentimientos, para determinar el estado de ánimo dentro de los mensajes de una red social o un foro de aprendizaje.
- Realizar la implementación del análisis de texto con la integración de una base de sinónimos como la de WordNet.
- Realizar un sistema de reconocimiento de audio para procesar el lenguaje natural.

BIBLIOGRAFÍA:

About WordNet. (2015). Retrieved January 5, 2015, from http://wordnet.princeton.edu/

Alcántara Plá, M. (2007). *Introducción al análisis de estructuras lingüísticas en corpus, Manuel Alcántara Plá*. Madrid: UAM Ediciones. Retrieved from http://www.inicios.es/introduccion-linguistica-corpus/

Allen, J. (1995). *Natural Language Understanding*. Redwood City: Benjamin-Cummings Publishing Co., Inc.

Amazon. (2014). Amazon Mechanical Turk. Retrieved September 10, 2015, from http://docs.aws.amazon.com/es_es/AWSMechTurk/latest/AWSMechanicalTurkGettingStartedGuide/amt-gsg.pdf

Apache OpenNLP - Welcome to Apache OpenNLP. (2010). Retrieved September 10, 2015, from http://opennlp.apache.org/

Apache OpenNLP Development Community. (2014). Apache OpenNLP Developer Documentation. Retrieved August 16, 2015, from https://opennlp.apache.org/documentation/1.6.0/manual/opennlp.html

Argomedo Pflücker, K. C., & Cóndor Ruiz, A. T. (2014). *Diseño de una propuesta de sistema inteligente utilizando procesamiento de lenguaje natural (PLN) para reconocimiento de mensajes extorsivos*. Retrieved from http://www.inf.unitru.edu.pe/revistas/2014/7.pdf

Baeza Yates, R., & Ribeiro Neto, B. (1999). *Modern Information Retrieval*. New York: ACM Press.

Brun, R. E., & Senso, J. A. (2004). Minería textual. Retrieved September 1, 2015, from http://eprints.rclis.org/11491/1/Artmineriapdf.pdf

Cabral Morales, M. (2006). Intervención grupal en las emociones desencadenadas por el rumor en zonas en riesgo de desastre. *PsicologiaCientifica.com*, *8*, 3. Retrieved from http://www.psicologiacientifica.com/zonas-riesgo-desastre-intervencion-grupal/

Caicedo Carvajal, J. M. (2012). Spanish POS Tagger OpenNLP Models. Retrieved September 21, 2015, from http://cavorite.com/labs/nlp/opennlp-models-es/

Covington, M. A. (1994). *Natural Language Processing for Prolog Programmers*. New Jersey: Prentice Hall. Retrieved from http://www.covingtoninnovations.com/books/NLPPP.pdf

Diéguez, Á. S. (2008). Extracción de información. Retrieved March 18, 2015, from https://sites.google.com/site/recuperarorganizarinformacion/extracci%C3%B3ndeinformaci%C3%B3n

Equihua, S. (2014). Data & Text Mining - Infotecarios. Retrieved November 9, 2015, from http://www.infotecarios.com/data-text-mining/

Galicia Haro, S. N., & Gelbukh, A. (2007). *Investigaciones en análisis sintáctico para el español*. Instituto Politécnico Nacional. Dirección de Publicaciones. Retrieved from http://www.gelbukh.com/libro-investigaciones/LibroSint.pdf

Gallego, A. J. (2008). La jerarquía de Chomsky y la facultad del lenguaje: consecuencias para la variación y la evolución. *Teorema: Revista Internacional de Filosofía*, *27*(2), 47–60. Retrieved from http://dialnet.unirioja.es/servlet/articulo?codigo=2580734

Grishman, R. (1986). Computational Linguistics: an introduction. Cambridge University Press.

Indurkhya, N., & Damerau, F. J. (2010). *Handbook of Natural Language Processing, Second Edition*. CRC Press. Retrieved from https://books.google.com/books?hl=en&lr=&id=nK-QYHZ0-_gC&pgis=1

Lancaster, F. W. (1968). *Information Retrieval Systems: Characteristics, Testing and Evaluation*. New York.

López García, Á., & Gallardo Paúls, B. (2011). *Conocimiento y lenguaje* (Vol. 28). Universitat de València. Retrieved from https://books.google.com/books?id=hiaqpw7WsXkC&pgis=1

Manaris, B. Z., & Slator, B. M. (1996). *Interactive Natural Language Processing: Building on Success*. Computer, IEEE.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Retrieved November 9, 2015, from http://www-nlp.stanford.edu/IR-book/

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). *The Stanford CoreNLP Natural Language Processing Toolkit*. Retrieved from http://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf

Martí, M. A., & Taulé, M. (2011). La Academia y la investigación universitaria en las tecnologías de la lengua. Retrieved March 18, 2015, from https://docs.google.com/file/d/0B6N0v65RwfFSN1RBWGtWVmpLTXc/edit?pli=1

Microsoft. (2015). Conceptos de minería de datos. Retrieved September 9, 2015, from https://msdn.microsoft.com/es-es/library/ms174949(v=sql.120).aspx

Mitkov, R. (2005). *The Oxford Handbook of Computational Linguistics*. Oxford University Press.

Moreno Sandoval, A. (1998). *Lingüística Computacional. Introducción a los modelos simbólicos, estadísticos y biológicos*. Madrid: Editorial Sintesis.

Natural Language Toolkit — NLTK 3.0 documentation. (2015). Retrieved September 10, 2015, from http://www.nltk.org/index.html

Oatley, K. (1992). *Best Laid Schemes: The Psychology of the Emotions*. Cambridge University Press. Retrieved from https://books.google.com/books?id=H14npd9i_icC&pgis=1

Ortega Rodríguez, F. J. (2008). *STR. Un generador de etiquetadores supervisados basado en*

TextRank. Escuela Técnica Superior de Ingeniería Informática. Retrieved from https://www.lsi.us.es/docs/doctorado/memorias/MemoPerInvOrtegaRodriguezFcoJavier.pdf

Padró, L. (2013). FreeLing User Manual. Retrieved May 13, 2015, from http://nlp.lsi.upc.edu/freeling/doc/userman/html/

Pérez Hernández, M. C. (2002). Explotación de los córpora textuales informatizados para la creación de bases de datos terminológicas basadas en el conocimiento, *18*(1139-8736). Retrieved from http://elies.rediris.es/elies18/index.html

Real Academia Española. (2015). Términos lingüísticos | Real Academia Española. Retrieved April 13, 2015, from http://www.rae.es

Tan, A. (1999). *Text Mining: The state of the art and challenges, Proc. of the Workshop Knowledge Discovery from advanced Databases*. Pennsylvania.

Thakker, D., Osman, T., & Lakin, P. (2009). *GATE JAPE Grammar Tutorial*. Nottingham: GATE. Retrieved from https://gate.ac.uk/sale/thakker-jape-tutorial/GATE JAPE manual.pdf

The Eagles Lexicon Interest Group. (2011). ETIQUETAS EAGLES. Retrieved April 13, 2015, from http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html

The Stanford Natural Language Processing Group. (2014). The Stanford Parser: A statistical parser. Retrieved August 11, 2015, from http://nlp.stanford.edu/software/lex-parser.shtml

The Stanford NLP (Natural Language Processing) Group. (2012). Retrieved September 10, 2015, from http://nlp.stanford.edu/software/corenlp.shtml

Torruella, J., & Llisterri, J. (1999). Diseño de corpus textuales y orales. Retrieved February 4, 2015, from http://latel.upf.edu/traductica/lc/material/torruella_llisterri_99.pdf

van Rijsbergen, C. J. (1979). Information Retrieval. Retrieved January 6, 2015, from http://www.dcs.gla.ac.uk/Keith/Preface.html

Vilares Ferro, J. (2005). *Aplicaciones del procesamiento del lenguaje natural en la recuperación de información en español*. Universidade Da Coruña. Retrieved from http://coleweb.dc.fi.udc.es/cole/library/ps/Vil2005a.pdf

W3C. (2015). Extensible Markup Language (XML). Retrieved November 9, 2015, from http://www.w3.org/XML/

ANEXOS

ANEXO 1: Stanford CoreNLP ejemplo completo - POS Tagger

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package ec.edu.utpl.analizador;

import edu.stanford.nlp.tagger.maxent.MaxentTagger;


/**
 *
 * @author Mario
 */

public class Tagger {

    public static void main(String[] args){

        //Iniciar el tagger
        MaxentTagger tagger = new MaxentTagger("taggers/spanish.tagger");

        // Ejemplo de una cadena
        String ejemplo = "Esta es una oración de prueba";
        // Cadena
        String taggeo = tagger.tagString(ejemplo);
        // Presentamos el resultado
        System.out.println(taggeo);

    }


}
```

ANEXO 2: Stanford CoreNLP ejemplo completo - Reconocedor de la entidad

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package ec.edu.utpl.analizador;

import edu.stanford.nlp.ie.AbstractSequenceClassifier;
import edu.stanford.nlp.ie.crf.CRFClassifier;
import edu.stanford.nlp.ling.CoreAnnotations;
import edu.stanford.nlp.ling.CoreLabel;
import java.io.IOException;
import java.util.List;

/**
 *
 * @author Mario
 */

public class Ner {

    public static void main(String[] args) throws ClassNotFoundException,
ClassCastException, IOException {


        AbstractSequenceClassifier<CoreLabel> classifier =
CRFClassifier.getClassifier("edu/stanford/nlp/models/ner/spanish.ancora.distsim.
s512.crf.ser.gz");
        // Ejemplo de una cadena
        String ejemplo = "En el día más brillante, en la noche más oscura, El
mal no escapará a mi vista. Que aquellos que adoran al mal, Teman mi poder: ¡LA
LUZ DE LINTERNA VERDE!";

        List<List<CoreLabel>> out = classifier.classify(ejemplo);

        for (List<CoreLabel> oracion : out) {
            for (CoreLabel palabra : oracion) {
                System.out.print("('"+palabra.word() + "','"
+palabra.get(CoreAnnotations.AnswerAnnotation.class) + ") ");
            }
            System.out.println();
        }



    }
}
```

ANEXO 3: Stanford CoreNLP ejemplo completo - Parser

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package ec.edu.utpl.analizador;

import edu.stanford.nlp.ling.CoreLabel;
import edu.stanford.nlp.parser.lexparser.LexicalizedParser;
import edu.stanford.nlp.process.CoreLabelTokenFactory;
import edu.stanford.nlp.process.PTBTokenizer;
import edu.stanford.nlp.process.Tokenizer;
import edu.stanford.nlp.process.TokenizerFactory;
import edu.stanford.nlp.trees.Tree;
import java.io.StringReader;
import java.util.List;

/**
 *
 * @author Mario
 */

public class Parser {

    public static void main(String[] args) {

        LexicalizedParser lexpars =
LexicalizedParser.loadModel("edu/stanford/nlp/models/lexparser/spanishPCFG.ser.g
z","-maxLength", "80", "-retainTmpSubcategories");


        String texto = "El reino canta muy bien.";

        TokenizerFactory<CoreLabel> tokenizerFactory = PTBTokenizer.factory(new
CoreLabelTokenFactory(), "");
        Tokenizer<CoreLabel> token = tokenizerFactory.getTokenizer(new
StringReader(texto));
        List<CoreLabel> rawPalabras = token.tokenize();

        Tree parse = lexpars.apply(rawPalabras);
        parse.pennPrint();


    }

}
```

ANEXO 4: OpenNLP ejemplo completo - Detección de oraciones

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package ec.edu.utpl.nlp.sentdetect;

import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.Charset;

import opennlp.tools.sentdetect.SentenceDetectorME;
import opennlp.tools.sentdetect.SentenceModel;
import opennlp.tools.sentdetect.SentenceSampleStream;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;
import opennlp.tools.util.TrainingParameters;

/**
 *
 * @author Mario
 */

public class SentecesDetector {

    private static final String RUTA_TEXTO =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\sentdetector\\
el_filosofo_autodidactico.txt";
    private static final String MODELDIR =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\sentdetector\\
es-sent.bin";
    private static final String TEXTO = "En el día más brillante, en la noche
más oscura, El mal no escapará a mi vista. Que aquellos que adoran al mal, Teman
mi poder: ¡LA LUZ DE LINTERNA VERDE!";

    public static void main(String[] args) throws IOException {

        System.out.println("Prueba");
         evaluar();
        System.out.println("Entrenamiento");
         entrenamiento();

    }

    private static void evaluar() throws FileNotFoundException {
        System.out.println("Evaluar...");

        InputStream modelIn = new FileInputStream(MODELDIR);
        try {
            SentenceModel model = new SentenceModel(modelIn);
```

```
            SentenceDetectorME sentenceDetector = new SentenceDetectorME(model);
            String sentences[] = sentenceDetector.sentDetect(TEXTO);

            for (int i = 0; i < sentences.length; i++) {
                System.out.println("[" + (i + 1) + "]" + "==> " + sentences[i]);
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (modelIn != null) {
                try {
                    modelIn.close();
                } catch (IOException e) {
                }
            }
        }
    }

    private static void entrenamiento() throws IOException {
        System.out.println("Entrenar...");

        Charset charset = Charset.forName("UTF-8");
        ObjectStream lineStream = new PlainTextByLineStream(new
FileInputStream(RUTA_TEXTO), charset);
        ObjectStream sampleStream = new SentenceSampleStream(lineStream);

        SentenceModel model;

        try {
            model = SentenceDetectorME.train("es", sampleStream, true, null,
TrainingParameters.defaultParams());

        } finally {
            sampleStream.close();
        }

        OutputStream modelOut = null;
        try {

            modelOut = new BufferedOutputStream(new FileOutputStream(MODELDIR));
            model.serialize(modelOut);

        } finally {
            if (modelOut != null) {
                modelOut.close();
            }
        }
    }
}
```

ANEXO 5: OpenNLP ejemplo completo - Tokenizador

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package ec.edu.utpl.nlp.sentdetect;

import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.Charset;
import opennlp.tools.tokenize.TokenSampleStream;
import opennlp.tools.tokenize.Tokenizer;
import opennlp.tools.tokenize.TokenizerME;
import opennlp.tools.tokenize.TokenizerModel;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;
import opennlp.tools.util.TrainingParameters;

/**
 *
 * @author Mario
 */

public class Tokens {

    private static final String RUTA_TEXTO =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\tokens\\token_
train.txt";
    private static final String MODELDIR =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\tokens\\es-
token.bin";
    private static final String TEXTO = "En el día más brillante, en la noche
más oscura, El mal no escapará a mi vista. Que aquellos que adoran al mal, Teman
mi poder: ¡LA LUZ DE LINTERNA VERDE!";

    public static void main(String[] args) throws IOException {
        System.out.println("Entrenamiento");
        entrenamiento();
        System.out.println("Tokenización: ");
        evaluar();

    }

    private static void evaluar() throws FileNotFoundException {

        InputStream modelIn = new FileInputStream(MODELDIR);
        try {
            TokenizerModel model = new TokenizerModel(modelIn);
            Tokenizer tokenizer = new TokenizerME(model);
            String tokens[] = tokenizer.tokenize(TEXTO);
```

```
            for (int i = 0; i < tokens.length; i++) {
                System.out.println("[" + (i + 1) + "]" + "==> " + tokens[i]);
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (modelIn != null) {
                try {
                    modelIn.close();
                } catch (IOException e) {
                }
            }
        }
    }

    private static void entrenamiento() throws IOException {
        System.out.println("Entrenar...");
        Charset charset = Charset.forName("UTF-8");
        ObjectStream lineStream = new PlainTextByLineStream(new
FileInputStream(RUTA_TEXTO), charset);
        ObjectStream sampleStream = new TokenSampleStream(lineStream);
        TokenizerModel model;
        try {
            model = TokenizerME.train("es", sampleStream, true,
TrainingParameters.defaultParams());
        } finally {
            sampleStream.close();
        }
        OutputStream modelOut = null;
        try {
            modelOut = new BufferedOutputStream(new FileOutputStream(MODELDIR));
            model.serialize(modelOut);

        } finally {
            if (modelOut != null) {
                modelOut.close();
            }
        }
    }

}
```

ANEXO 6: OpenNLP ejemplo completo - POS Tagging

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package ec.edu.utpl.nlp.sentdetect;

import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.StringReader;
import java.nio.charset.Charset;
import opennlp.tools.postag.POSModel;
import opennlp.tools.postag.POSSample;
import opennlp.tools.postag.POSTaggerME;
import opennlp.tools.postag.WordTagSampleStream;
import opennlp.tools.tokenize.WhitespaceTokenizer;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;
import opennlp.tools.util.TrainingParameters;

/**
 *
 * @author Mario
 */

public class Tagging {

    private static final String RUTA_TEXTO =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\POS\\es-
train.txt";
    private static final String MODELDIR =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\POS\\SpanishPO
S.bin";
    private static final String TEXTO = "En el día más brillante, en la noche
más oscura, El mal no escapará a mi vista. Que aquellos que adoran al mal, Teman
mi poder: ¡LA LUZ DE LINTERNA VERDE!";

    public static void main(String[] args) throws IOException {

      System.out.println("Entrenamiento");
      entrenamiento();
      System.out.println("POS Tagging: ");
      evaluar();

   }

    private static void evaluar() throws FileNotFoundException {
        InputStream modelIn = new FileInputStream(MODELDIR);
        try {
            POSModel model = new POSModel(modelIn);
```

```
            POSTaggerME tagger = new POSTaggerME(model);
            ObjectStream<String> lineStream = new PlainTextByLineStream(new
StringReader(TEXTO));
            String line;
            while ((line = lineStream.read()) != null) {
                String whitespaceTokenizerLine[] =
WhitespaceTokenizer.INSTANCE.tokenize(line);
                String[] tags = tagger.tag(whitespaceTokenizerLine);
                for (int i = 0; i < tags.length; i++) {
                    System.out.println("[" + (i + 1) + "]" + "==> " + "( " +
tags[i] + " => " + whitespaceTokenizerLine[i] + " )");
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (modelIn != null) {
                try {
                    modelIn.close();
                } catch (IOException e) {
                }
            }
        }
    }

    private static void entrenamiento() throws IOException {
        System.out.println("Entrenar...");
        POSModel model = null;
        InputStream dataIn = null;
        Charset charset = Charset.forName("UTF-8");
        try {
            dataIn = new FileInputStream(RUTA_TEXTO);
            ObjectStream<String> lineStream = new PlainTextByLineStream(dataIn,
"UTF-8");
            ObjectStream<POSSample> sampleStream = new
WordTagSampleStream(lineStream);
            model = POSTaggerME.train("es", sampleStream,
TrainingParameters.defaultParams(), null, null);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (dataIn != null) {
                try {
                    dataIn.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        OutputStream modelOut = null;
        try {
            modelOut = new BufferedOutputStream(new FileOutputStream(MODELDIR));
            model.serialize(modelOut);
        } catch (IOException e) {
            // Failed to save model
            e.printStackTrace();
        } finally {
            if (modelOut != null) {
```

```
                try {
                    modelOut.close();
                } catch (IOException e) {
                    // Failed to correctly save model.
                    // Written model might be invalid.
                    e.printStackTrace();
                }
            }

        }
    }
}
```

ANEXO 7: OpenNLP ejemplo completo - Reconocedor de la entidad

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package ec.edu.utpl.nlp.sentdetect;

import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.Charset;
import java.util.Arrays;
import java.util.Collections;
import opennlp.tools.namefind.NameFinderME;
import opennlp.tools.namefind.NameSample;
import opennlp.tools.namefind.NameSampleDataStream;
import opennlp.tools.namefind.TokenNameFinderModel;
import opennlp.tools.tokenize.TokenSampleStream;
import opennlp.tools.tokenize.WhitespaceTokenizer;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;
import opennlp.tools.util.Span;
import opennlp.tools.util.TrainingParameters;

/**
 *
 * @author Mario
 */

public class Ner {

    private static final String MODELDIR =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\ner\\en-ner-
person.bin";
    //private static final String TEXTO = "En el día más brillante, en la noche
más oscura, El mal no escapará a mi vista. Que aquellos que adoran al mal, Teman
mi poder: ¡LA LUZ DE LINTERNA VERDE! .";
    private static final String TEXTO = "Mario Correa , 23 años";
    private static final String RUTA_TEXTO =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\ner\\train-
ner.txt";

    public static void main(String[] args) throws IOException {
        System.out.println("Entrenamiento");
        entrenamiento();
        System.out.println("NER: ");
        evaluar();

    }

    private static void entrenamiento() throws IOException {
```

```
        System.out.println("Entrenar...");
        Charset charset = Charset.forName("UTF-8");
        ObjectStream lineStream = new PlainTextByLineStream(new
FileInputStream(RUTA_TEXTO), charset);
        ObjectStream<NameSample> sampleStream = new
NameSampleDataStream(lineStream);
        TokenNameFinderModel model;
        try {
            model = NameFinderME.train("es", "nombre", sampleStream,
Collections.<String, Object>emptyMap());
        } finally {
            sampleStream.close();
        }
        OutputStream modelOut = null;
        try {
            modelOut = new BufferedOutputStream(new FileOutputStream(MODELDIR));
            model.serialize(modelOut);
        } finally {
            if (modelOut != null) {
                modelOut.close();
            }
        }
    }

    private static void evaluar() throws FileNotFoundException, IOException {
        InputStream modelIn = new FileInputStream(MODELDIR);
        try {
            TokenNameFinderModel model = new TokenNameFinderModel(modelIn);
            NameFinderME nameFinder = new NameFinderME(model);
            String whitespaceTokenizerLine[] =
WhitespaceTokenizer.INSTANCE.tokenize(TEXTO);
            Span nameSpans[] = nameFinder.find(whitespaceTokenizerLine);
            for (Span s : nameSpans) {
                System.out.println(s.toString());
            }
            System.out.println(" " +
Arrays.toString(Span.spansToStrings(nameSpans, whitespaceTokenizerLine)));
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (modelIn != null) {
                try {
                    modelIn.close();
                } catch (IOException e) {
                }
            }
        }
    }
}
```

ANEXO 8: OpenNLP ejemplo completo - Parser

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package ec.edu.utpl.nlp.sentdetect;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.Arrays;
import opennlp.tools.cmdline.parser.ParserTool;
import opennlp.tools.parser.Parse;
import opennlp.tools.parser.ParserFactory;
import opennlp.tools.parser.ParserModel;

/**
 *
 * @author Mario
 */

public class Parser {

    private static final String MODELDIR =
"C:\\Users\\Usuario\\Documents\\NetBeansProjects\\Tesis\\OpenNLP\\parsing\\es-
parser-chunking.bin";
    private static final String TEXTO = "El reino canta muy bien";

    public static void main(String[] args) throws FileNotFoundException,
IOException {
        System.out.println("NER: ");
            evaluar();

    }

    private static void evaluar() throws FileNotFoundException {
        InputStream modelIn = new FileInputStream(MODELDIR);
        try {
            ParserModel model = new ParserModel(modelIn);
            opennlp.tools.parser.Parser parser = ParserFactory.create(model);
            Parse topParses[] = ParserTool.parseLine(TEXTO,
(opennlp.tools.parser.Parser) parser, 1);
            for (Parse p : topParses){
                p.show();
            }
            System.out.println();

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (modelIn != null) {
                try {
                    modelIn.close();
                } catch (IOException e) {
```

```
            }
        }
    } } }
```

ANEXO 9: Etiquetas Eagles

Esta tabla es un resumen de la nomenclatura de etiquetas eagles que se encuentran en el capítulo 2, en el punto 2.7.

Etiquetas			
Código	**Descripción**	**Código**	**Descripción**
Adjetivos		**Intersecciones**	
AO	Adjetivo Ordinal	**I**	Intersección
AQ	Adjetivo Calificativo	**Preposiciones**	
Adverbios		**SP**	Preposición, Adposición
RG	Adverbio General	**Puntuación**	
RN	Adverbio Negativo	**Faa**	Inicia exclamación (¡)
Determinantes		**Fat**	Cierra exclamación (¡)
DD	Determinante Demostrativo	**Fc**	Coma (,)
DP	Determinante Posesivo	**Fca**	Inicia corchetes ([)
DT	Determinante Interrogativo	**Fct**	Cierra corchetes (])
DE	Determinante Exclamativo	**Fd**	Dos puntos (:)
DI	Determinante Indefinido	**Fe**	Comillas (“)
DA	Determinante Articulo	**Fg**	Guión Alto (-)
Nombres		**Fh**	Slash (/)
NC	Nombre Común	**Fia**	Inicia interrogación (¿)
NP	Nombre Propio	**Fit**	Cierra Interrogación (?)
Verbos		**Fla**	Inicia Llaves ({)
VMI	Verbo Principal, Indicativo	**Flt**	Cierra Llaves (})
VMS	Verbo Principal, Subjuntivo	**Fp**	Punto (.)
VMM	Verbo Principal, Imperativo	**Fpa**	Inicia paréntesis (“(“)
VMN	Verbo Principal, Infinitivo	**Fpt**	Cierra paréntesis (“)”)
VMG	Verbo Principal, Gerundio	**Fra**	Inicia comillas angulares («)
VMP	Verbo Principal, Participativo	**Frc**	Cierra comillas angulares (»)
VAI	Verbo Auxiliar, Indicativo	**Fs**	Tres puntos (…)
VAS	Verbo Auxiliar, Subjuntivo	**Ft**	Porcentaje (%)
VAM	Verbo Auxiliar, Imperativo	**Fx**	Punto y coma (;)
VAN	Verbo Auxiliar, Infinitivo	**Fz**	Otros signos (+,_,*,=)
VAG	Verbo Auxiliar, Gerundio	**Números**	
VAP	Verbo Auxiliar, Participativo	**Z**	Cifra
VSI	Verbo Semiauxiliar, Indicativo	**Fecha/Hora**	
VSM	Verbo Semiauxiliar, Subjuntivo	**W**	Fecha y hora
VSN	Verbo Semiauxiliar, Infinitivo	**Desconocido**	
VSG	Verbo Semiauxiliar, Gerundio	**X**	Desconocido
VSP	Verbo Semiauxiliar, Participativo		
Pronombres			
PP	Pronombre Personal		
PD	Pronombre Demostrativo		
PX	Pronombre Posesivo		
PI	Pronombre Indefinido		
PT	Pronombre Interrogativo		
PR	Pronombre Relativo		
PE	Pronombre Exclamativo		
Conjunciones			

CC	Conjunción Coordinada		
CS	Conjunción Subordinada		

ANEXO 10: Palabras ocupadas en los diccionarios para realizar el etiquetado.

Aburrimiento					
abandonar	dejadez	Enojar	inapetencia	modorra	Rollo
abandono	Dejar	enojo	incomodar	modorrear	Rozar
Abulia	desalientizar	extenuación	incomodidad	molestar	somnolencia
Aburrido	desaliento	extenuar	incordio	molestia	somnolienta
aburrimiento	desganar	fastidiar	indiferencia	molimiento	Sopor
adormecer	desgane	fastidio	indolencia	monotonia	Tedio
adormecimiento	desidia	fatiga	insensibilidad	monotonía	Tibieza
Aflicción	desidiar	fatigar	insensibilizar	neutralidad	Tostón
Afligir	desinterés	hartar	insistencia	neutralizar	
Agobiar	desinterés	hartazgo	insistir	pena	
agobio	desinterezar	hartura	languidez	penar	
agotamiento	disgustar	hastiar	lasitud	pereza	
agotar	disgusto	hastio	lata	pesadez	
amargura	empalagar	hastío	aletargar	pesadumbre	
amodorramiento	empalago	impasibilidad	letargo	repugnancia	
apatía	enfadar	impasibilizar	malestar	repugnar	
cansancio	enfado	apetecer	matraca	roce	
Angustia					
ansiedad	desespero	indecisión	pesimismo	Vacilación	
arrepentimiento	desfalleciemiento	indisposición	preocupación	Zozobra	
azoramiento	desmayo	inquietud	quebranto		
codicia	dificultad	inseguridad	recelo		
concoja	disgusto	intranquilidad	remordimiento		
congoja	disturbio	intranquilizar	sacrificio		
conmoción	dolor	jaleo	sentimiento		
desaliento	duda	malestar	sinsabor		
desánimo	enojo	melancolía	sospecha		
desasosiego	esfuerzo	molestia	sufrimiento		
desazón	exasperación	nerviosismo	suplicio		
desconsolar	excitación	nostalgia	tormento		
desconsuelo	fatiga	padecimiento	tortura		
desdicha	follón	pena	trabajo		
deseo	impaciencia	penalidad	tribulación		
desesperación	incertidumbre	pesadumbre	tristeza		
desesperanza	incomodidad	pesar	turbación		
Ansiedad					
abatido	Azoramiento	desfallecido	Histeria	Nervio	receloso
afán	codicia	desmayado	impaciente	nervios	remorder
aflijido	congojado	desolado	incertidumbre	neurastenia	sacrificio
agita	conmocionado	desvelo	incomodo	neurosis	sinsabor
agobiado	consternado	difícil	indeciso	nostalgia	sospecho
agoniza	contrariado	disgustado	indisposición	padecer	sufrir
ahogo	desaliento	disturbio	inseguro	pasión	suplicio
alarmado	desasosiego	dolor	insomnio	pena	susto
albor	desazón	esfuerzo	intranquilidad	pesadumbre	temor
amargado	desconsolado	esquizofrenia	intrigado	pesar	tensión

angustiado	desconsuelo	estrés	irrita	pésimo	tormenta
anhelado	desdicha	exaspera	malestar	preocupa	tortura
ansia	desea	excitado	manía	presión	tribia
apremia	deseo	exita	melancolía	presionad	turba
arrepentido	desesperado	fatigado	miedo	prisa	urge
asustado	desespera	fatiga	mortifica	psicópata	
ayuda	desesperante	hipocondría	necesidad	quebrantado	
Aburrimiento					
abandonar	dejadez	enojar	inapetencia	Modorra	rollo
abandono	dejar	enojo	incomodar	modorrear	rozar
abulia	desalientizar	extenuación	incomodidad	molestar	somnolencia
aburrido	desaliento	extenuar	incordio	molestia	somnolienta
aburrimiento	desganar	fastidiar	indiferencia	molimiento	sopor
adormecer	desgane	fastidio	indolencia	monotonia	tedio
adormecimiento	desidia	fatiga	insensibilidad	monotonía	tibieza
aflicción	desidiar	fatigar	insensibilizar	Neutralidad	tostón
afligir	desinterés	hartar	insistencia	Neutralizar	
agobiar	desinterés	hartazgo	insistir	Pena	
agobio	desinterezar	hartura	languidez	Penar	
agotamiento	disgustar	hastiar	lasitud	Pereza	
agotar	disgusto	hastio	lata	Pesadez	
amargura	empalagar	hastío	aletargar	Pesadumbre	
amodorramiento	empalago	impasibilidad	letargo	Repugnancia	
apatía	enfadar	impasibilizar	malestar	Repugnar	
cansancio	enfado	apetecer	matraca	Roce	
Confusión					
aclara	confuso	desorientado	fallo	indispuesto	perplejo
acláre	consterna	disparate	falseamiento	inexacto	perturba
adónde	consulta	disputa	fárrago	inopia	precipita
alocamiento	consultarle	distraído	follón	inquieto	pregunta
ambiguo	cuándo	disturbio	gazapo	inteligible	qué
anomalía	cuánto	divaga	generalidad	irreflexible	quién
argucia	defecto	dónde	gresca	irregular	retruécano
ataque	deforme	duda	ignoran	irresoluble	revuelta
atarugamiento	desacierto	embrollo	impetuoso	jolgorio	rodeo
atolondrado	desasosiego	enigma	impreciso	ligereza	secreto
aturdido	desbarajuste	enredo	imprudente	lio	tergiversad
barullo	desconcierto	equivocación	incomprensión	locura	torpe
bulla	desconocido	errad	inconsciente	maraña	tumulto
bullicio	descuido	error	indeciso	mezcolanza	turbación
caos	desliz	escandalo	indefinido	misterio	yerro
cómo	desorden	estrépito	indetermina	ofusco	
complica	desorganiza	evasivo	indique	pendencia	
Frustración					
abortar	decepcionar	desengañar	equívoco	hundimiento	planchar
aborto	defecto	desengaño	errata	hundir	revéz
caer	defectuar	desesperanza	error	inexactitud	yerrar
caída	derrota	desesperanzar	estropiciar	infortunio	yerro
calamidad	derrotar	desgracia	estropicio	malograr	
chasco	desaciertar	desgraciar	fallar	malogro	

chasquear	desacierto	desilución	fallo	mentir	
chazcar	desairar	desilucionar	falsear	mentira	
chazco	desaire	disgustar	falsedad	olvidar	
confundir	desastre	disgusto	falta	olvido	
confusión	descalabrar	disparate	faltar	patinar	
contrariar	descalabro	encarmientizar	fiazco	patinazo	
contrariedad	descuidar	encarmiento	fracasar	percance	
cortar	descuido	equivocación	fracaso	percanzar	
corte	desencantar	equivocar	frustración	perdidad	
decepción	desencanto	equívocar	frustrado	plancha	
Simpatía					
aceptación	bondad	encantamiento	penetración	vinculación	
acoplamiento	cabezada	encantar	persuación	voluntad	
acuerdo	capacidad	encanto	persuadir		
ademán	captación	enganchar	preciosidad		
adhesión	capturar	entender	predilección		
adhesionar	caricia	entendimiento	predisposición		
adoración	cariño	estima	proclividad		
afabilidad	cercanía	exquisitez	propensar		
afecto	coincidencia	familiaridad	propensión		
afectuosidad	coincidenciar	fascinación	proximidad		
afición	compatibilidad	fascinar	querencia		
afinidad	compenetración	gancho	ratificación		
agraciar	compenetrar	genuflexión	razón		
agudeza	comprensión	gracia	relación		
alcance	concomitancia	halago	relacionar		
alucinación	concordancia	hechizar	resignación		
amabilidad	concordializar	hechizo	reverencia		
amistad	conexión	hermosura	saludo		
amoldamiento	conformar	humanidad	seducción		
amor	conformidad	inclinación	seducir		
analogía	consanguinidad	inclinar	semejanza		
apasionamiento	consenso	ingenio	sencillez		
apegar	consentimiento	instinto	sentimiento		
apego	consentizar	intelecto	similitud		
apoyar	coquetear	inteligencia	simpatía		
apoyo	coquetería	interés	simpatico		
aprecio	cordialidad	juicio	sociabilidad		
aprobación	correlación	lindeza	solidaridad		
aproximación	correspondencia	lucidez	sombrerazo		
aproximar	delicia	magia	sugestión		
armonía	deslumbramiento	maravilla	sumisión		
atracción	devoción	maravillar	talento		
atractivo	dilección	mimo	tendencia		
atraer	discernimiento	paciencia	tendenciar		
avenencia	embelesco	parecido	ternura		
avidez	embrujar	parentesco	tolerancia		
belleza	embrujo	pasión	transigencia		

ANEXO 11: Listado de Stopwords

a	algunos	aquí	cada
aca	alla	arriba	casi
acá	allá	arribaabajo	casi
actualmente	alli	aseguró	cerca
acuerdo	allí	asi	cierta
adelante	alrededor	así	ciertas
ademas	ambos	atras	cierto
además	ampleamos	aun	ciertos
adrede	antano	aún	cinco
afirmó	antaño	aunque	claro
agregó	ante	ayer	comentó
ahi	anterior	b	comingo
ahí	antes	bajo	como
ahiah	añadió	bastante	cómo
ahora	apenas	bien	con
ajena	aproximadamente	breve	conmigo
ajenas	aquel	buen	conocer
ajeno	aquél	buena	conseguimos
ajenos	aquella	buenas	conseguir
al	aquélla	bueno	considera
algo	aquellas	buenos	consideró
algun	aquéllas	c	consigo
algún	aquello	c#	consigue
alguna	aquellos	c++	consiguen
algunas	aquéllos	cabe	consigues
alguno	aqui	cada	contigo

contra
cosas
creo
cual
cuál
cuales
cuáles
cualquier
cualquiera
cualquieras
cuan
cuán
cuando
cuándo
cuanta
cuánta
cuantas
cuántas
cuanto
cuánto
cuantos
cuántos
cuatro
cuenta
d
da
dado
dan
dar
de
debajo
debe
deben
debido
decir
dejar
dejó
del
delante
demas
demás
demasiada
demasiadas
demasiado
demasiados
dentro
deprisa
desde
despacio
despues
después
detras
detrás
dia
día
dias
días
dice
dicen
dicho
dieron
diferente
diferentes
dijeron
dijo
dio
donde
dónde
dos
durante
e
ejemplo
el
él
ella
ellas
ello
ellos
embargo
empleais
emplean
emplear
empleas
empleo
en
encima
encuentra
enfrente
enseguida
entonces
entre
era
eramos
eran
eras
eres
es
esa
ésa
esas
ésas
ese
ése
eso
esos
ésos
esta
está

ésta
estaba
estaban
estado
estados
estais
estamos
estan
están
estar
estará
estas
éstas
este
éste
esto
estos
éstos
estoy
estuvo
etc
eva
EVA
ex
excepto
existe
existen
explicó
expresó
f
fin
final
fue
fuera
fueron
fui
fuimos
g
general
gran
grandes
gueno
h
ha
haber
habia
había
habían
habla
hablan
habrá
hace
haceis
hacemos
hacen
hacer
hacerlo
haces
hacia
haciendo
hago
han
hasta
hay
haya
he
hecho
hemos
hicieron
hizo
hola
horas
hoy
hubo
i
igual
incluso
indicó
informo
informó
ing
ing.
ingeniera
ingeniero
intenta
intentais
intentamos
intentan
intentar
intentas
intento
ir
j
jamas
jamás
java
junto
juntos
k
l
la
lado
largo
las
le
lejos
les
llegó

lleva
llevar
lo
los
luego
lugar
m
mal
manera
manifestó
mas
más
mayor
me
mediante
medio
mejor
mencionó
menos
menudo
mi
mí
mia
mía
mias
mías
mientras
mio
mío
mios
míos
mis
misma
mismas
mismo
mismos
modo
momento
mucha
muchas
muchisima
muchísima
muchísimas
muchísimo
muchísimos
mucho
muchos
muy
n
nada
nadie
ni
ningun
ningún
ninguna
ningunas
ninguno
ningunos
no
nos
nosotras
nosotros
nuestra
nuestras
nuestro
nuestros
nueva
nuevas
nuevo
nuevos
nunca
nunguno
o
ocho
os
otra
otras
otro
otros
p
pais
país
para
parece
parecer
parte
partir
pasada
pasado
peor
pero
pesar
php
poca
pocas
poco
pocos
podeis
podemos
poder
podrá
podrán
podria
podría
podriais
podriamos
podrian
podrían

podrias
poner
por
porque
porque
posible
primer
primera
primero
primeros
principalmente
pronto
propia
propias
propio
propios
proximo
próximo
próximos
pudo
pueda
puede
pueden
puedo
pues
q
qeu
que
qué
quedó
queremos
querer
quien
quién
quienes
quiénes
quienesquiera
quienquiera
quiere
quiza
quizá
quizas
quizás
r
raras
realizado
realizar
realizó
repente
respecto
s
sabe
sabeis
sabemos
saben
saber
sabes
salvo
se
sé
sea
sean
segun
según
segunda
segundo
seis
señaló
ser
sera
será
serán
sería
si
sí
sido
siempre
siendo
siete
sigue
siguiente
sin
sín
sino
so
sobre
sois
sola
solamente
solas
solo
sólo
solos
somos
son
soy
soyos
sr
sra
sres
sta
su
supuesto
sus
suya
suyas
suyo
suyos

t
tal
tales
tambien
también
tampoco
tan
tanta
tantas
tanto
tantos
tarde
te
temprano
tendrá
tendrán
teneis
tenemos
tener
tenga
tengo
tenía
tenido
tercera
ti
tiempo
tiene
tienen
toda
todas
todavia
todavía
todo
todos
tomar
total
trabaja
trabajais
trabajamos
trabajan
trabajar
trabajas
trabajo
tras
trata
través
tres
tu
tú
tus
tuvo
tuya
tuyas
tuyo
tuyos
u
ud
última
últimas
ultimo
último
últimos
un
una
unas
uno
unos
usa
usais
usamos
usan
usar
usas
uso
usted
ustedes
v
va
vais
valor
vamos
van
varias
varios
vaya
veces
ver
verdad
verdadera
verdadero
vez
vosotras
vosotros
voy
vuestra
vuestras
vuestro
vuestros
w
x
xq
y
ya
yo
z

ANEXO 11: Corpus usados en pruebas

Archivo denominado Datos_1.

Corpus	
Oración 1	Tampoco descartaron que la suspensión de las negociaciones lleve a Israel a provocar el aumento de la tensión en el sur de Líbano, donde ocupa una franja territorial de unos mil kilómetros cuadrados desde 1978.
Tagging	Tampoco_**RG** descartaron_**VMIS3P0** que_**CS** la_**DA0FS0** suspensión_**NCFS000** de_**SPS00** las_**DA0FP0** negociaciones_**NCFP000** lleve_**VMSP3S0** a_**SPS00** Israel_**NP00000** a_**SPS00** provocar_**VMN0000** el_**DA0MS0** aumento_**NCMS000** de_**SPS00** la_**DA0FS0** tensión_**NCFS000** en_**SPS00** el_**DA0MS0** sur_**NCMS000** de_**SPS00** Líbano_**NP00000** ,_**Fc** donde_**PR000000** ocupa_**VMIP3S0** una_**DI0FS0** franja_**NCFS000** territorial_**AQ0CS0** de_**SPS00** unos_**DI0MP0** mil_**Z** kilómetros_**NCMP000** cuadrados_**VMP00PM** desde_**SPS00** 1978_**Z** ._**Fp**
Total tokens	37
Oración 2	Trias admitió que, si bien el pacto de gobierno con el PP ha sido una "buena experiencia" en algunos aspectos, como en la política económica o en los traspasos de competencias para Cataluña, "no estamos satisfechos" de la "falta importante de sensibilidad" del PP en algunos aspectos sociales y en la concepción de Estado plurinacional.
Tagging	Trias_**NP00000** admitió_**VMIS3S0** que_**CS** ,_**Fc** si_**CS** bien_**RG** el_**DA0MS0** pacto_**NCMS000** de_**SPS00** gobierno_**NCMS000** con_**SPS00** el_**A0MS0** pp_**NP00000** ha_**VAIP3S0** sido_**VSP00SM** una_**DI0FS0** "_**Fe** buena_**AQ0FS0** experiencia_**NCFS000** "_**Fe** en_**SPS00** algunos_**DI0MP0** aspectos_**NCMP000** ,_**Fc** como_**CS** en_**SPS00** la_**DA0FS0** política_**NCFS000** económica_**AQ0FS0** o_**CC** en_**SPS00** los_**DA0MP0** traspasos_**NCMP000** de_**SPS00** competencias_**NCFP000** para_**SPS00** Cataluña_**NP00000** ,_**Fc** "_**Fe** no_**RN** estamos_**VAIP1P0** satisfechos_**VMP00PM** "_ **Fe** de_**SPS00** la_**DA0FS0** "_**Fe** falta_**NCFS000** importante_**AQ0CS0** de_**SPS00** sensibilidad_**NCFS000** "_**Fe** de_**SPS00** el_**DA0MS0** PP_**NP00000** en_**SPS00** algunos_**DI0MP0** aspectos_**NCMP000** sociales_**AQ0CP0** y_**CC** en_**SPS00** la_**DA0FS0** concepción_**NCFS000** de_**SPS00** Estado_**NP00000** plurinacional_**AQ0CS0** ._**Fp**
Total tokens	66

Oración 3	Agentes de viaje, tour-operadores, representantes de líneas aéreas y personas relevantes de la vida cultural y política española y canaria asistieron a la velada en la que actuaron más de 50 artistas
Tagging	Agentes_**NCCP000** de_**SPS00** viaje_**NCMS000** ,_**Fc** tour_**NCMS000** -_**Fg** operadores_**NCMP000** ,_**Fc** representantes_**NCCP000** de_**SPS00** líneas_**NCFP000** aéreas_**AQ0FP0** y_**CC** personas_**NCFP000** relevantes_**AQ0CP0** de_**SPS00** la_**DA0FS0** vida_**NCFS000** cultural_**AQ0CS0** y_**CC** política_**NCFS000** española_**AQ0FS0** y_**CC** canaria_**AQ0FS0** asistieron_**VMIS3P0** a_**SPS00** la_**DA0FS0** velada_**NCFS000** en_**SPS00** la_**DA0FS0** que_**PR0CN000** actuaron_**VMIS3P0** más_**RG** de_**SPS00** 50_**Z** artistas_**NCCP000** ._**Fp**
Total tokens	37
Oración 4	Alaui expresó su voluntad de "luchar contra estas irregularidades, en las que participan, deliberada o inconscientemente, algunos delegados o directores de establecimientos escolares".
Tagging	Alaui_**NP00000** expresó_**VMIS3S0** su_**DP3CS0** voluntad_**NCFS000** de_**SPS00** "_**Fe** luchar_**VMN0000** contra_**SPS00** estas_**DD0FP0** irregularidades_**NCFP000** ,_**Fc** en_**SPS00** las_**DA0FP0** que_**PR0CN000** participan_**VMIP3P0** ,_**Fc** deliberada_**VMP00SF** o_**CC** inconscientemente_**RG** ,_**Fc** algunos_**DI0MP0** delegados_**NCMP000** o_**CC** directores_**NCMP000** de_**SPS00** establecimientos_**NCMP000** escolares_**AQ0CP0** "_**Fe** ._**Fp**
Total tokens	29
Total tokens corpus	169

Archivo denominado Datos_2.

Corpus	
Oración 1	El Consejo, integrado por 41 países, se dedica al fomento de la democracia y los derechos humanos.
Tagging	El_**DA0MS0** Consejo_**NP00000** ,_**Fc** integrado_**VMP00SM** por_**SPS00** 41_**Z** países_**NCMP000** ,_**Fc** se_**P00CN000** dedica_**VMIP3S0** a_**SPS00** el_**DA0MS0** fomento_**NCMS000** de_**SPS00** la_**DA0FS0** democracia_**NCFS000** y_**CC** los_**DA0MP0** derechos_**NCMP000** humanos_**AQ0MP0** ._**Fp**
Total tokens	21
Oración 2	Los partidarios de privar a la delegación rusa de su derecho de voto, algo que nunca se ha hecho en el Consejo, reconocieron la necesidad de mantener el diálogo con Moscú, pero dijeron que el Consejo perdería su credibilidad si no actuaba.
Tagging	Los_**DA0MP0** partidarios_**NCMP000** de_**SPS00** privar_**VMN0000** a_**SPS00** la_**DA0FS0** delegación_**NCFS000** rusa_**AQ0FS0** de_**SPS00** su_**DP3CS0** derecho_**NCMS000** de_**SPS00** voto_**NCMS000** ,_**Fc** algo_**PI0CS000** que_**PR0CN000** nunca_**RG** se_**P00CN000** ha_**VAIP3S0** hecho_**VMP00SM** en_**SPS00** el_**DA0MS0** Consejo_**NP00000** ,_**Fc** reconocieron_**VMIS3P0** la_**DA0FS0** necesidad_**NCFS000** de_**SPS00** mantener_**VMN0000** el_**DA0MS0** diálogo_**NCMS000** con_**SPS00** Moscú_**NP00000** ,_**Fc** pero_**CC** dijeron_**VMIS3P0** que_**CS** el_**DA0MS0** Consejo_**NP00000** perdería_**VMIC3S0** su_**DP3CS0** credibilidad_**NCFS000** si_**CS** no_**RN** actuaba_**VMII3S0** ._**Fp**
Total tokens	46
Oración 3	El ministro ruso de asuntos exteriores, quien canceló una rueda de prensa tras la votación, declaró en la Asamblea que la "fase activa" de la operación "antiterrorista" acabará "pronto", pero advirtió de que la intervención tendrá que ir "hasta su término" para poder restablecer "los derechos" en Chechenia.

Tagging	El_DA0MS0 ministro_NCMS000 ruso_AQ0MS0 de_SPS00 asuntos_NCMP000 exteriores_AQ0CP0 ,_Fc quien_PR0CS000 canceló_VMIS3S0 una_DI0FS0 rueda_NCFS000 de_SPS00 prensa_NCFS000 tras_SPS00 la_DA0FS0 votación_NCFS000 ,_Fc declaró_VMIS3S0 en_SPS00 la_DA0FS0 Asamblea_NCFS000 que_PR0CN000 la_DA0FS0 "_Fe fase_NCFS000 activa_AQ0FS0 "_Fe de_SPS00 la_DA0FS0 operación_NCFS000 "_Fe antiterrorista_AQ0CS0 "_Fe acabará_VMIF3S0 "_Fe pronto_RG "_Fe ,_Fc pero_CC advirtió_VMIS3S0 de_SPS00 que_CS la_DA0FS0 intervención_NCFS000 tendrá_VMIF3S0 que_CS ir_VMN0000 "_Fe hasta_SPS00 su_DP3CS0 término_NCMS000 "_Fe para_SPS00 poder_VMN0000 restablecer_VMN0000 "_Fe los_DA0MP0 derechos_NCMP000 "_Fe en_SPS00 Chechenia_NCFS000 ._Fp
Total tokens	62
Oración 4	"Pero no es momento de hacer más valoraciones, salvo la repulsa más rotunda ante un atentado de estas características", subrayó.
Tagging	"**_Fe** Pero**_CC** no**_RN** es**_VSIP3S0** momento**_NCMS000** de**_SPS00** hacer**_VMN0000** más**_RG** valoraciones**_NCFP000** ,**_Fc** salvo**_SPS00** la**_DA0FS0** repulsa**_NCFS000** más**_RG** rotunda**_AQ0FS0** ante**_SPS00** un**_DI0MS0** atentado**_NCMS000** de**_SPS00** estas**_DD0FP0** características**_NCFP000** "**_Fe** ,**_Fc** subrayó**_VMIS3S0** .**_Fp**
Total tokens	25
Total tokens corpus	154

Archivo denominado Datos_3.

Corpus	
Oración 1	Para esta profesora "la originalidad de esta investigación radica en el hecho de que no hay estudios previos que nos digan porqué los ruidos de algunos coches nos resultan más molestos que otros, siempre se ha actuado de forma física sobre los sistemas de escape".
Tagging	Para_**SPS00** esta_**DD0FS0** profesora_**NCFS000** "_**Fe** la_**DA0FS0** originalidad_**NCFS000** de_**SPS00** esta_**DD0FS0** investigación_**NCFS000** radica_**VMIP3S0** en_**SPS00** el_**DA0MS0** hecho_**NCMS000** de_**SPS00** que_**PR0CN000** no_**RN** hay_**VMIP3S0** estudios_**NCMP000** previos_**AQ0MP0** que_**PR0CN000** nos_**PP1CP000** digan_**VMSP3P0** porqué_**NCMS000** los_**DA0MP0** ruidos_**NCMP000** de_**SPS00** algunos_**DI0MP0** coches_**NCMP000** nos_**PP1CP000** resultan_**VMIP3P0** más_**RG** molestos_**AQ0MP0** que_**CS** otros_**PI0MP000** ,_**Fc** siempre_**RG** se_**P00CN000** ha_**VAIP3S0** actuado_**VMP00SM** de_**SPS00** forma_**NCFS000** física_**AQ0FS0** sobre_**SPS00** los_**DA0MP0** sistemas_**NCMP000** de_**SPS00** escape_**NCMS000** "_**Fe** ._**Fp**
Total tokens	49
Oración 2	El viceministro explicó que su Gobierno apoya que los tribunales chilenos "asuman plenamente el papel histórico que les corresponde en la búsqueda de la verdad y la justicia para las numerosas víctimas de violaciones a sus derechos humanos y libertades fundamentales".
Tagging	El_**DA0MS0** viceministro_**NCMS000** explicó_**VMIS3S0** que_**CS** su_**DP3CS0** Gobierno_**NP00000** apoya_**VMIP3S0** que_**CS** los_**DA0MP0** tribunales_**NCMP000** chilenos_**AQ0MP0** "_**Fe** asuman_**VMSP3P0** plenamente_**RG** el_**DA0MS0** papel_**NCMS000** histórico_**AQ0MS0** que_**PR0CN000** les_**PP3CPD00** corresponde_**VMIP3S0** en_**SPS00** la_**DA0FS0** búsqueda_**NCFS000** de_**SPS00** la_**DA0FS0** verdad_**NCFS000** y_**CC** la_**DA0FS0** justicia_**NCFS000** para_**SPS00** las_**DA0FP0** numerosas_**AQ0FP0** víctimas_**NCFP000** de_**SPS00** violaciones_**NCFP000** a_**SPS00** sus_**DP3CP0** derechos_**NCMP000** humanos_**AQ0MP0** y_**CC** libertades_**NCFP000** fundamentales_**AQ0CP0** "_**Fe** ._**Fp**
Total tokens	44
Oración 3	El concurso convocado para determinar el autor de la escultura ya fue

	convocado y en noviembre se cerrará el plazo de admisión de bocetos.
Tagging	El_**DA0MS0** concurso_**NCMS000** convocado_**VMP00SM** para_**SPS00** determinar_**VMN0000** el_**DA0MS0** autor_**NCMS000** de_**SPS00** la_**DA0FS0** escultura_**NCFS000** ya_**RG** fue_**VSIS3S0** convocado_**VMP00SM** y_**CC** en_**SPS00** noviembre_**W** se_**P00CN000** cerrará_**VMIF3S0** el_**DA0MS0** plazo_**NCMS000** de_**SPS00** admisión_**NCFS000** de_**SPS00** bocetos_**NCMP000** ._**Fp**
Total tokens	25
Oración 4	Goirizelaia reprochó al PNV, EA, IU y CDN que no respaldaran esta jornada de movilización y se contenten con "declaraciones pomposas sin hacer nada", por lo que les pidió "que se definan si están con la sociedad de Herria y la defensa de los presos o no".
Tagging	Goirizelaia_**NP00000** reprochó_**VMIS3S0** a_**SPS00** el_**DA0MS0** PNV_**NP00000** ,_**Fc** EA_**NP00000** ,_**Fc** IU_**NP00000** y_**CC** CDN_**NP00000** que_**CS** no_**RN** respaldaran_**VMSI3P0** esta_**DD0FS0** jornada_**NCFS000** de_**SPS00** movilización_**NCFS000** y_**CC** se_**P00CN000** contenten_**VMSP3P0** con_**SPS00** "_**Fe** declaraciones_**NCFP000** pomposas_**AQ0FP0** sin_**SPS00** hacer_**VMN0000** nada_**PI0CS000** "_**Fe** ,_**Fc** por_**SPS00** lo_**DA0NS0** que_**PR0CN000** les_**PP3CPD00** pidió_**VMIS3S0** "_**Fe** que_**PR0CN000** se_**P00CN000** definan_**VMSP3P0** si_**CS** están_**VAIP3P0** con_**SPS00** la_**DA0FS0** sociedad_**NCFS000** de_**SPS00** Herria_**NP00000** y_**CC** la_**DA0FS0** defensa_**NCCS000** de_**SPS00** los_**DA0MP0** presos_**NCMP000** o_**CC** no_**RN** "_**Fe** ._**Fp**
Total tokens	56
Total tokens corpus	174

Archivo denominado Datos_4.

Corpus	
Oración 1	Durante toda la semana ha estado en tratamiento médico con un nuevo sistema de recuperación mediante ordenador, que parece haber sido efectivo, por lo que cabe la posibilidad de que juegue ante el conjunto vitoriano.
Tagging	Durante_**SPS00** toda_**DI0FS0** la_**DA0FS0** semana_**NCFS000** ha_**VAIP3S0** estado_**VAP00SM** en_**SPS00** tratamiento_**NCMS000** médico_**AQ0MS0** con_**SPS00** un_**DI0MS0** nuevo_**AQ0MS0** sistema_**NCMS000** de_**SPS00** recuperación_**NCFS000** mediante_**SPS00** ordenador_**NCMS000** ,_**Fc** que_**PR0CN000** parece_**VMIP3S0** haber_**VMN0000** sido_**VSP00SM** efectivo_**AQ0MS0** ,_**Fc** por_**SPS00** lo_**DA0NS0** que_**PR0CN000** cabe_**VMIP3S0** la_**DA0FS0** posibilidad_**NCFS000** de_**SPS00** que_**CS** juegue_**VMSP3S0** ante_**SPS00** el_**DA0MS0** conjunto_**NCMS000** vitoriano_**AQ0MS0** ._**Fp**
Total tokens	38
Oración 2	El lanzador Ajete, medallista de plata en Sydney, aseguró a EFE que no jugaría este sábado, pero apoyaría a sus compañeros.
Tagging	El_**DA0MS0** lanzador_NCMS000 Ajete_**NP00000** ,_**Fc** medallista_**NCCS000** de_**SPS00** plata_**NCFS000** en_**SPS00** Sydney_**NP00000** ,_**Fc** aseguró_**VMIS3S0** a_**SPS00** EFE_**NP00000** que_**CS** no_**RN** jugaría_**VMIC3S0** este_**DD0MS0** sábado_**W** ,_**Fc** pero_**CC** apoyaría_**VMIC3S0** a_**SPS00** sus_**DP3CP0** compañeros_**NCMP000** ._**Fp**
Total tokens	25
Oración 3	"Lo he pasado mal, pero es de las pocas alegrías que me he llevado este año, ojalá me lleve más, podemos llegar muy lejos; esto te hace olvidar situaciones difíciles y ahora disfruta uno de estos momentos", subrayó Alfonso.
Tagging	"_**Fe** Lo_**PP3CNA00** he_**VAIP1S0** pasado_**VMP00SM** mal_**RG** ,_**Fc** pero_**CC** es_**VSIP3S0** de_**SPS00** las_**DA0FP0** pocas_**DI0FP0** alegrías_**NCFP000** que_**PR0CN000** me_**PP1CS000** he_**VAIP1S0** llevado_**VMP00SM** este_**DD0MS0** año_**NCMS000** ,_**Fc** ojalá_**RG** me_**PP1CS000** lleve_**VMSP3S0** más_**RG** ,_**Fc** podemos_**VMIP1P0**

	llegar_**VMN0000** muy_**RG** lejos_**RG** ;_**Fx** esto_**PD0NS000** te_**PP2CS000** hace_**VMIP3S0** olvidar_**VMN0000** situaciones_**NCFP000** difíciles_**AQ0CP0** y_**CC** ahora_**RG** disfruta_**VMIP3S0** uno_**PI0MS000** de_**SPS00** estos_**DD0MP0** momentos_**NCMP000** "_**Fe** ,_**Fc** subrayó_**VMIS3S0** Alfonso_**NP00000** ._**Fp**
Total tokens	47
Oración 4	"Hice eso para limpiar la imagen de Brasil en el exterior", habría afirmado Alves.
Tagging	"_**Fe** Hice_**VMIS1S0** eso_**PD0NS000** para_**SPS00** limpiar_**VMN0000** la_**DA0FS0** imagen_**NCFS000** de_**SPS00** Brasil_**NP00000** en_**SPS00** el_**DA0MS0** exterior_**NCMS000** "_**Fe** ,_**Fc** habría_**VAIC1S0** afirmado_**VMP00SM** Alves_**NP00000** ._**Fp**
Total tokens	18
Total tokens corpus	128

ANEXO 12: Manual de Usuario

INTRODUCCIÓN

El aplicativo ayuda a realizar el análisis de textos en español y el etiquetado de sentimientos. No cuenta con login, así que cualquier persona puede tener acceso a la misma.

Uso de la aplicación

Para acceder a este aplicativo, descargamos el archivo jar de la siguiente dirección:

- https://www.dropbox.com/s/dtpwuxq3lfh3q8s/PLNDemo.jar?dl=0

Ya descargado y ubicado en un lugar para facilitar su uso, procedemos a ejecutarlo, lo único que debe hacer es darle doble click al jar y se presentará la ventana principal de la aplicación, como lo vemos en la figura 1:

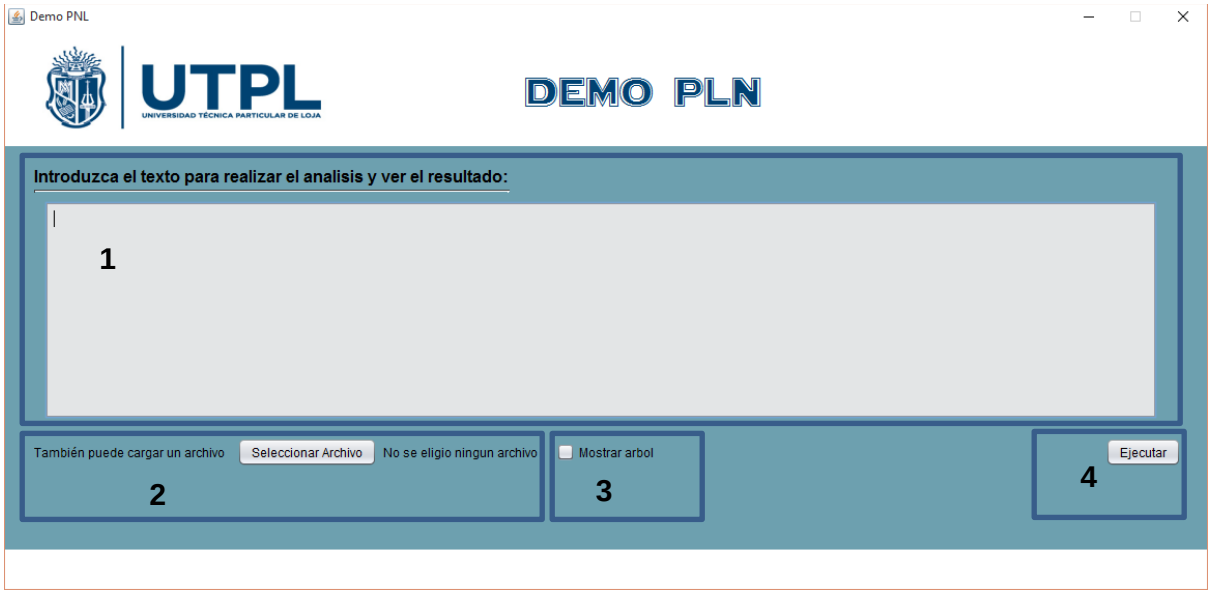

Figura 1. Ventana principal de la aplicación

En la figura 1, podemos encontrar las diferentes formas en las que se puede realizar el proceso de análisis del texto. La primera (1), se escribirá el texto directamente en la aplicación, y la segunda (2) subir un archivo de tipo txt. También nos encontramos con una opción para presentar un árbol (3) y con el botón para ejecutar (4)

De cualquiera de las dos formas que se ocupe se llegará al mismo resultado, pero antes de ejecutar el proceso, se tiene la opción de marcar la casilla que dice “Mostrar árbol”. Una vez que

se ejecute se presentara la segunda ventana de la aplicación, es la que se presenta en la figura 2. Para este ejemplo se ocupó el segundo método.

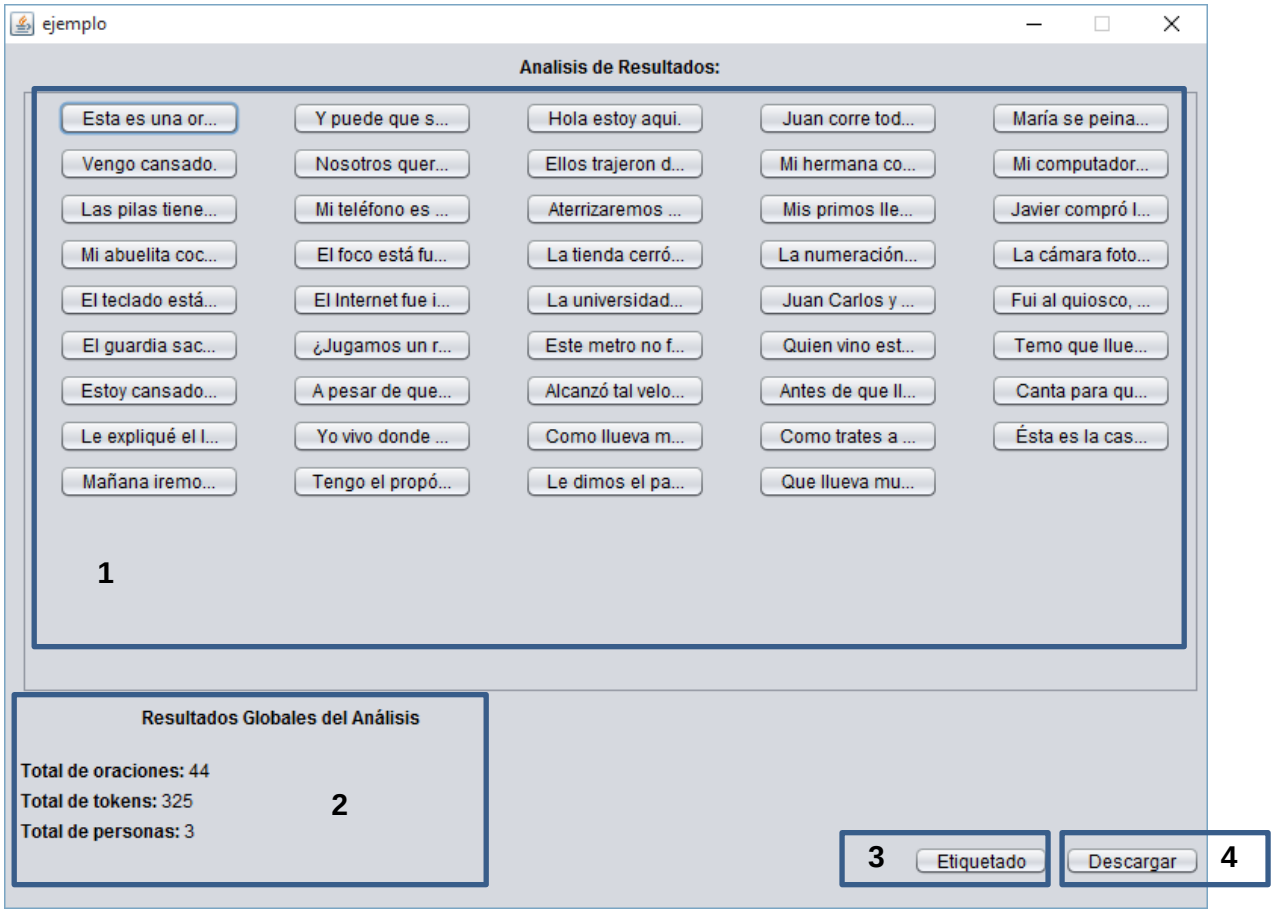

Figura 2. Presentación del análisis del texto

En la figura 2, podemos ver el texto ya separado por oraciones (1), resultados globales del análisis (2), botón para realizar el etiquetado (3), y un botón para descargar los datos (4).

Botón de etiquetado

Al presionar este botón se presentara una pequeña ventana con un mensaje, para seleccionar el tipo de archivo en el que se lo desea descargar, las opciones que se tiene son: arff y txt. Esta ventana la podemos ver en la figura 3.

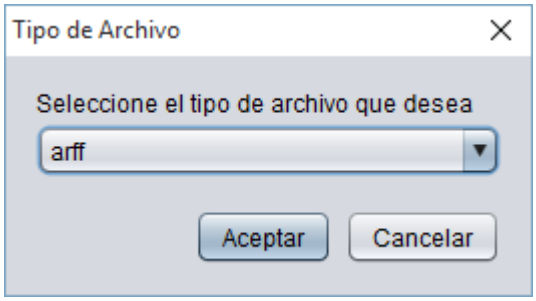

Figura 3. Etiquetado – Seleccionar tipo de archivo

Una vez escogido el tipo del archivo, Aceptamos esto y nos procede a descargar un archivo, a este lo podemos encontrar en la misma ruta en la que tenemos el ejecutable de esta aplicación.

Botón de descargar

Al hacer click en este botón se creara un archivo de tipo json, en la ruta donde está el ejecutable de la aplicación y solo presentara un mensaje de que la descarga se realizó correctamente. Como lo vemos en la figura 4.

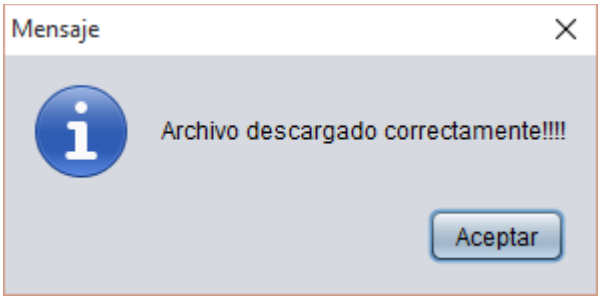

Figura 4. Descarga – Mensaje correcto de la descarga

Resultados globales del análisis

En esta parte se presenta datos globales del texto analizado como son la cantidad de oraciones encontradas, el total de tokens que existen y cuantas personas se han encontrado en el texto.

Panel de oraciones

En esta sección encontramos un conjunto de botones, los cuales corresponde a la cantidad de oraciones encontradas en el texto analizado. En la figura 2 podemos ver este panel (2).

Al presionar uno de estos botones nos presentara el análisis de la oración seleccionada. Como lo vemos en la figura 5.

Figura 5. Análisis de una oración

Si se seleccionó la opción de “Marcar árbol” en la ventana principal, aparecer un botón que dice “Ver Árbol” (1), caso contrario este no aparecerá. Al hacer click nos presenta una nueva ventana, como la de la figura 6.

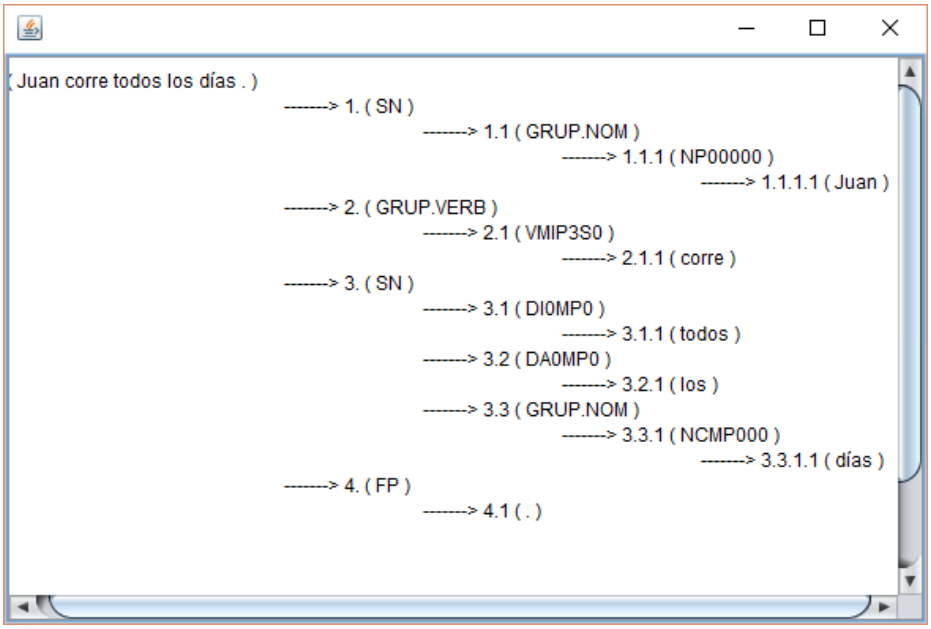

Figura 6. Árbol binario de la estructura de la oración.

ANEXO 13: Paper sobre la investigación

Adaptación de una herramienta de procesamiento de lenguaje natural para el etiquetado de sentimientos y el análisis del lenguaje en español

Mario Correa, Ing. Prisila Valdiviezo
Profesional en formación, Universidad Técnica Particular de Loja
Docente del DCCE, Universidad Técnica Particular de Loja
Autores para correspondencia: *mfcorrea@utpl.edu.ec, pmvaldiviezo@utpl.edu.ec*

ABSTRACT

The objective of the following title investigation was to perform an adaptation tool of natural language processor and also the labeling of emotions in Spanish, which allows to create a systematic and linguistic analysis of the text or a virtual environment, in order to establish an emotional and grammar pattern of each word that has been analyzed in order to develop a natural language.

Nowadays there are several programs to turn the processing natural language into a great help (PNL) but finally, after a practice analysis we manage to conclude with the use of OpenNLP because it consumes a less quantity of resources and the process is released in a less period of time. OpenNLP is a collection of projects distributed under the license of open code, developed in JAVA, which offers the following tools: Tokenize detection of sentences, entity recognizer and part-of-speech tagging.

In order to obtain the data, first we separated the text in sentences. Each phrase was divided in tokens in order to assign them a part-of-speech tagging. This was the proposal by the group EAGLES for the European languages, which also includes the Spanish language, Also, for the token we can assign a tag emotions like: boredom, anguish, anxiety or concern, confusion, frustration or sympathy.

Finally, a lot of tools are available in Spanish which are limited; therefore I found it convenient to add up a tool with better functions for the benefit of new investigations.

Keywords: PLN, token, linguistic analysis, OpenNLP, feelings, tags.

RESUMEN

El objetivo de este documento fue realizar la adaptación de una herramienta de Procesamiento de Lenguaje Natural y el Etiquetado de Sentimientos en Español, basados en el análisis sistemático y lingüístico de texto.

En la actualidad existen varios programas que ayudan a realizar un procesamiento de lenguaje natural (PLN), sin embargo en este trabajo se utiliza OpenNLP debido a las ventajas que presenta como: Consume una menor cantidad de recursos y realiza el procesamiento en menor tiempo. OpenNLP una colección de proyectos distribuidos bajo licencia de código abierto, desarrollado en Java, que ofrece las siguientes herramientas: Tokenizador, detección de oraciones, reconocedor de la entidad y etiquetado gramatical.

Para la obtención de los datos, primero se separó el texto en oraciones. Cada frase fue dividida en tokens para asignarle una etiqueta gramatical. Ésta etiqueta gramatical fue la propuesta por el grupo EAGLES para los idiomas europeos, que incluye el idioma español. Además, al token se le asignó un etiquetado de emociones como: aburrimiento, angustia, ansiedad o preocupación, confusión, frustración y simpatía.

Finalmente, muchas de las herramientas que están disponibles en español son limitadas, por lo tanto se creyó conveniente implementar un instrumento con mayores funcionalidades para el beneficio de nuevas investigaciones.

Palabras clave: PLN, token, OpenNLP, sentimientos, etiquetas, EAGLES.

1. INTRODUCCIÓN

Este trabajo se enfoca en la adaptación de una herramienta de Procesamiento de Lenguaje Natural que permita analizar textos en español a través de árboles sintácticos, identificando la categoría de cada palabra, por ejemplo sustantivo, verbo, adjetivo, etc.

1.1. PROCESAMIENTO DE LENGUAJE NATURAL

El Procesamiento de Lenguaje Natural nace en 1960, como una subárea de la Inteligencia Artificial y la Lingüística, con la finalidad de estudiar los problemas derivados del lenguaje natural.
En un principio tuvo una gran aceptación y éxito pero cuando se lo llevó a la práctica en campos no controlados y con vocabularios genéricos, empezaron a ocurrir problemas generados por la falta de comprensión de las maquinas del lenguaje natural.
(Covington, 1994) El Procesamiento de lenguaje natural es el uso de computadoras para entender lenguajes humanos (naturales) como inglés, francés y japonés, esto no quiere decir que el computador tenga pensamientos, sentimientos y conocimientos humanizados, sino que el dispositivo pueda reconocer y usar información expresada en lenguaje humano.

El PLN se ocupa en algunos campos como los expresados a continuación:

- Lingüística
- Ciencias de la computación
- Análisis Lingüístico
- Lenguaje
- Lenguaje Formal
- Comprensión del lenguaje
- Generación de textos
- Gramáticas Formales
- Matemática
- Neurociencia
- Definiciones empleadas en las gramáticas formales

La lingüística ha aportado grandes conocimientos sobre las lenguas naturales las cuales se pueden estructurar en algunos niveles como indica en la tabla 1:

Tabla 1. Niveles de conocimiento (Manaris & Slator, 1996)

	Características del nivel de conocimiento lingüístico	
Nivel	**Declarativo**	**Procedural**
Fonológico	Sonidos hablados	Formar morfemas
Morfológico	Unidades de las palabras, Palabras	Formar palabras, Derivar unidades de Significado.
Sintáctico	Funciones estructurales de palabras	Formar oraciones
Semántico	Significado independiente del contexto	Derivar significado de oraciones
Discurso	Funciones estructurales de oraciones	Formar diálogos
Pragmático	Significado dependiente del contexto	Derivar significado de oraciones relativo al discurso circundante

(Covington, 1994) Brinda una organización en niveles sobre el conocimiento lingüístico tales como:

- **Nivel Fonológico:** estudia como los sonidos son usados en el lenguaje, cada lenguaje posee sus fonemas. El nivel fonológico estudia: las realizaciones acústicas, por lo que solo aparece en los sistemas de reconocimiento de audio. Un poco separado del procesamiento del lenguaje natural ya que este analiza la onda del sonido.
- **Nivel Morfológico:** la morfología se encarga de la descripción de la estructura que posee cada palabra y de los procesos que la integran. Existen tres procesos para la formación de palabras que son: inflexión, derivación y composición, que son para evitar la expansión innecesaria de las palabras.
- **Nivel Sintáctico:** ofrece la construcción de las oraciones; Es el componente básico de los sistemas PLN que ayuda a reconocer las oraciones gramaticales y a asignarles una estructura. En 1957 Noam Chomsky fue el primero en hablar sobre esto.
- **Nivel Semántico:** la semántica ofrece el significado de la frase o su posible significado, acercando esta al lenguaje, analizándolas independientemente del contexto que posea.
- **Nivel Pragmático:** es el uso que se le da al contexto o al significado más allá de lo que dice la frase. Ayuda a comprender información que se encuentra sobreentendida, pero que no se llega a expresar en las frases u oraciones.
- **Nivel Discurso:** se almacena el conocimiento, que logra relacionar el significado de las oraciones aisladas e integrarlas, para formar una unidad más grande. Este conocimiento se ocupa de interpretar los pronombres anafóricos, etc. y es necesario para que en los sistemas exista conocimiento del contexto.

Para realizar actividades de procesamiento de lenguaje natural, primero se debe normalizar el texto de la siguiente manera:

- Tokenización de palabras.
- Normalización de la estructura de las palabras.
- Tokenización de las oraciones.

Antes de continuar con la tokenización o segmentación se debe revisar algunos conceptos como:

- **Lema:** forma de citar de una palabra. Un ejemplo, el lema de leíamos es leer
- **Forma de la palabra:** forma completa de una palabra.
- **Tipo:** clase de elementos o elemento de un vocabulario.
- **Token:** instancia de un tipo de texto dado.

En primer lugar está la tokenización responsable de dividir el texto de entrada en oraciones y palabras. La forma más sencilla de tokenizar es separar los caracteres alfabéticos en función al carácter del espacio.
Los errores que ocurren con la tokenización varían dependiendo del idioma, por ejemplo el alemán porque palabras compuestas se pueden escribir de corrido.
Seguidamente, tenemos la normalización de palabras, permite agrupar a las que poseen un mismo significado, por ejemplo: juego, juegos y Juegos, se tiene que hacer que todas estas palabras se representen por una sola que es juego.
Una de las acciones que se realiza en la normalización, es cambiar todo a minúsculas, pero en el idioma español existen algunas excepciones; las cuales se pueden solventar con la lematización.
Finalmente, se realiza la segmentación de las oraciones de un texto, analizando cada elemento individualmente; localizando algunos símbolos especiales como: ".", "!" o "?", estos identifican el fin de una oración, pero no siempre expresan ese significado. Para identificar cuando es el fin de una oración se usa un árbol de decisión, como el que se ilustra en la figura 1:

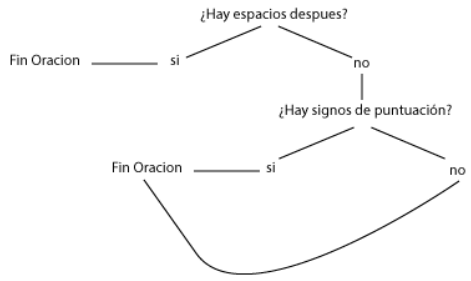

Figura 1. Árbol de decisión

Un árbol de decisión no es más que un conjunto de "if" y "else" anidados, los cuales permiten escoger lo que se evaluará en cada nodo. Sin embargo, las oraciones no tienen que ser evaluadas por un árbol de decisión, se puede ocupar otros clasificadores como las redes neuronales, la regresión logística, entre otros.

1.2. RECURSOS LINGÜÍSTICOS

("About WordNet," 2015) es una base de datos léxica en el idioma inglés, con sustantivos, verbos, adjetivos y adverbios, que se agrupan en conjuntos de sinónimos cognitivos (synsets), cada uno expresando un concepto distinto. Los synsets están vinculados entre sí por medio de relaciones conceptuales ya sean semánticas o léxicas.

Al igual que WordNet, EuroWordNet es una base de datos multilingüe con wordnets sobre varios idiomas europeos como el holandés, italiano, español, alemán, francés, checho y estonio. Están estructuradas de la misma forma que el americano. Estos wordnets representan un sistema de lenguaje único, se encuentran vinculadas a través de un índex al WordNet americano, y gracias a esta interconexión es posible buscar palabras parecidas en otro idioma.

A diferencia del WordNet original, la mayoría de los otros WordNets no están disponibles de forma gratuita.

1.3. CLASIFICACIÓN DE EMOCIONES

Es importante diferenciar que es un sentimiento y una emoción. Las emociones son respuestas que se dan a determinadas situaciones y se originan gracias a un impulso externo.
(Cabral Morales, 2006) dice, el psicólogo Robert Plutchik idéntico y clasifico las emociones que experimentan los seres humanos y animales, en 8 categorías básicas que motivan la conducta adaptativa, estas emociones son: Temor, Sorpresa, Tristeza, Repugnancia, Enojo, Esperanza, Dicha y Aceptación. Esta no es la única clasificación que existe.
En cambio, el sentimiento es un término que abarca más que solo sentir el estímulo. (Oatley, 1992) la define como: una experiencia afectiva en cierta medida agradable o desagradable, que supone una cualidad fenomenológica característica, que comprende tres sistemas de respuesta: cognitivo y subjetivo, conductual y expresivo y fisiológico y adaptativo.

2. SELECCIÓN Y ADAPTACIÓN DE LA HERRAMIENTA

Las herramientas como: Stanford CoreNLP, OpenNLP, Freeling, NLTK o GATE que fueron analizadas, se escogió OpenNLP, porque consume una menor cantidad de recursos y realiza el procesamiento en menor tiempo. En la tabla 2, encontramos un resumen de las características de esta.

Tabla 2. Características OpenNLP

OpenNLP	
Característica	**Descripción**
Modularidad	Si
Idioma	Español, Alemán, Danés, Portugués, Ingles, Holandés.
Lenguaje	Java
Licencia	GNU
Interfaz Grafica	No
Módulos	Segmentación de oraciones, tokenizador, etiquetado del texto, reconocedor de la entidad.

Para esta adaptación fue necesario la construcción de un módulo que permita: eliminar los “stopwords” y etiquetar los sentimientos.
Se usaron diccionarios de tipo raíz, por la gran cantidad de conjugaciones que pueden salir de una palabra. Los cuales fueron elaborados, por sinónimos de la emoción principal.
Estos diccionarios se crearon con ayuda de bases de datos léxicas como Wordnet, EuroWordnet o WordReference. Este último es un diccionario de sinónimos y antónimos.

.
Los requerimientos identificados para el desarrollo de este proyecto son los siguientes:

- Ingresar texto o corpus.
- Presentar los datos.
- Realizar el etiquetado de sentimientos
- Descargar los resultados.

En la figura 2 presentamos el esquema de los casos de uso para este proyecto. Estos casos de uso se crearon a base de los requerimientos que se presentaron previamente

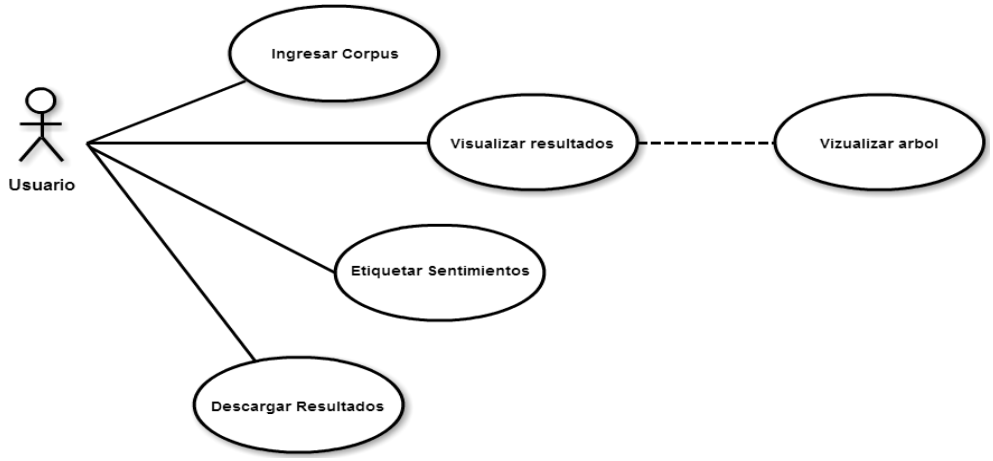

Figura 2. Diagrama general de casos de uso.

El resultado final lo podemos apreciar en la figura 3.

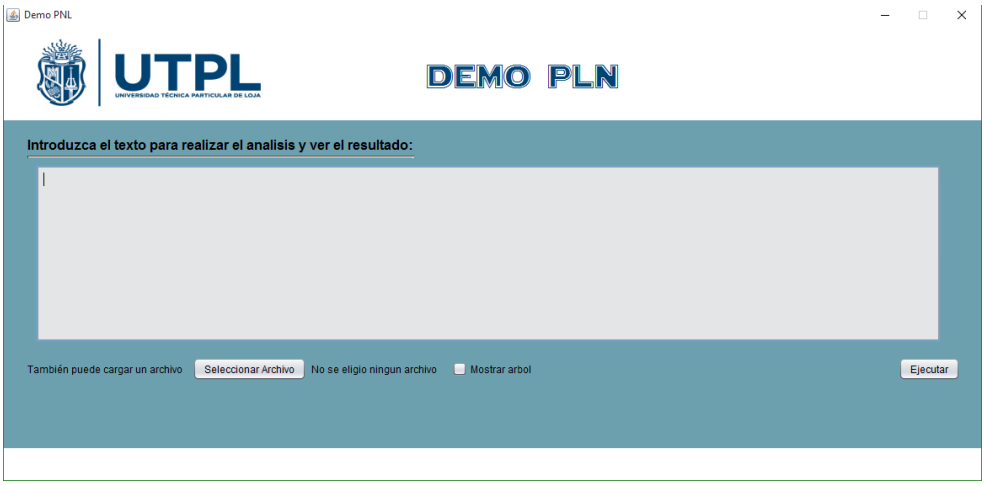

Figura 3. Ventana principal de la aplicación

Permite descargar los archivos en un formato cómodo para poder darle reutilización a los datos, para la parte de análisis del texto no permite descargar estos datos en un archivo .json, y si se trata del etiquetado de sentimiento se puede escoger entre descargar un archivo .txt o un .arff, el ultimo nos permitirá realizar un análisis de estos mensajes en Weka.

3. ANÁLISIS DE RESULTADOS DEL FUNCIONAMIENTO DE LA HERRAMIENTA

Se comprobó la precisión de los modelos, detección de oraciones, tokenización, etiquetado y reconocedor de la entidad. Estos resultados se detallan en la tabla 2.

Tabla 3. Comprobación de precisión de modelos

Criterios	**Precisión**	**Porcentaje**
Detección de oraciones	0,9743538	97%
Tokenización	0,9989394	100%
Etiquetado	0,9629507	96%
Ner	0,9195205	92%

Se validó de los resultados con un conjunto de datos que ya se encuentran etiquetados morfológicamente. Con ayuda de la herramienta de Freeling, formado cuatro corpus para realizar esta experimentación, los cuales se detallan en la tabla 3.

Tabla 4. Textos de prueba

Corpus	**Total de oraciones**	**Total de tokens**	**Total de etiquetas**
Datos_1	4	168	168
Datos_2	4	123	123
Datos_3	4	174	174
Datos_4	4	128	128

Las comparaciones respectivas, con los resultados obtenidos de la aplicación desarrollada y los obtenidos de estos corpus, están descritos en la figura 4.

Tabla 5. Análisis morfológico

Módulos	OpenNLP	
	Aciertos	Fallos
Detección de oraciones	100%	0%
Tokenizador	99,36%	0,64%
Etiquetador	96,25%	3,75%

En esta prueba, se ocupó dos bancos de mensajes de la red social del EVA de la modalidad abierta, del primer banco de mensajes se logró obtener 152 oraciones, un total de 5543 tokens, y 23 se encuentran catalogados como personas. Del segundo archivo se obtuvo 185 oraciones y 4237 tokens, encontrando en el mismo 11 que están catalogados como personas. A partir de estos tokens se procedió a crear un nuevo corpus con un etiquetado de sentimientos.

En la figura 4, podemos ver estos resultados de forma gráfica.

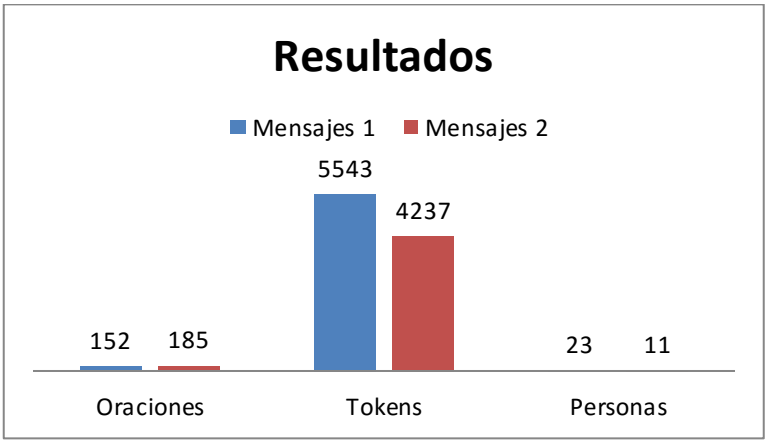

Figura 4. Gráfico de resultados generales del análisis

Las pruebas realizadas fueron para verificar el funcionamiento de la aplicación, se pueden hacer otras para verificar la calidad de resultados que realiza la aplicación.

4. CONCLUSIONES Y RECOMENDACIONES

Las herramientas que tienen una licencia open source (licencia GNU), ofrecen en algunos casos mayores beneficios que los de uso comercial, ya que poseen un desarrollo por parte de la comunidad, pero también pueden tener una mala documentación o una solo básica.

Es conveniente ocupar StopWord, porque permiten eliminar palabras que son irrelevantes que se encuentran en el texto, además no representan ninguna emoción.

Al trabajar con distintas herramientas existe el riesgo de que estas no se adapten, produciendo inestabilidad llegando a tener tiempos de respuesta más elevados y generar mayor cantidad de inconvenientes a la hora de depurar.

Se pueden encontrar varios recursos lingüísticos como WordNet o EuroWordNet que han sido tratados en otras investigaciones, las que se podría aprovechar para poder obtener mejores resultados, además de facilitarnos a la hora de buscar los sinónimos de la palabras con las que estamos trabajando.

La aplicación desarrollada, puede realizar el PLN de una gran cantidad de texto o corpus, pero mientras mayor sea la cantidad, el tiempo de análisis también será mayor.

Al trabajar con herramientas de licencia GNU, se debe revisar que éstas tengan la documentación necesaria, como manuales descriptivos, libros. Además de tener asistencia vía correos y de ser posible que posea una wiki.

No ocupar demasiadas herramientas de distintos proveedores porque pueden presentar dificultades al momento de realizar la integración de las mismas, pueden consumir más recursos de los necesarios.

Para trabajar con proyectos basados en el análisis de texto en idiomas, se debe hacer con ayuda de lingüistas o expertos en temas que tengan que ver con análisis de la gramática.

5. REFERENCIAS

About WordNet. (2015). Retrieved January 5, 2015, from http://wordnet.princeton.edu/

Alcántara Plá, M. (2007). *Introducción al análisis de estructuras lingüísticas en corpus, Manuel Alcántara Plá*. Madrid: UAM Ediciones. Retrieved from http://www.inicios.es/introduccion-linguistica-corpus/

Allen, J. (1995). *Natural Language Understanding*. Redwood City: Benjamin-Cummings Publishing Co., Inc.

Apache OpenNLP - Welcome to Apache OpenNLP. (2010). Retrieved September 10, 2015, from http://opennlp.apache.org/

Baeza Yates, R., & Ribeiro Neto, B. (1999). *Modern Information Retrieval*. New York: ACM Press.

Cabral Morales, M. (2006). Intervención grupal en las emociones desencadenadas por el rumor en zonas en riesgo de desastre. *PsicologiaCientifica.com, 8,* 3. Retrieved from http://www.psicologiacientifica.com/zonas-riesgo-desastre-intervencion-grupal/

Caicedo Carvajal, J. M. (2012). Spanish POS Tagger OpenNLP Models. Retrieved September 21, 2015, from http://cavorite.com/labs/nlp/opennlp-models-es/

Covington, M. A. (1994). *Natural Language Processing for Prolog Programmers*. New Jersey: Prentice Hall. Retrieved from http://www.covingtoninnovations.com/books/NLPPP.pdf

Fabra, U. P. (2012, June 5). El Procesamiento del Lenguaje Natural en la Recuperación de Información Textual y áreas afines. Universitat Pompeu Fabra. Retrieved from http://www.upf.edu/hipertextnet/numero-5/pln.html

Gelbukh, A., & Sidorov, G. (2006). *Procesamiento automático del español con enfoque en recursos léxicos grandes*. México DF.

Lancaster, F. W. (1968). *Information Retrieval Systems: Characteristics, Testing and Evaluation*. New York.

López García, Á., & Gallardo Paúls, B. (2011). *Conocimiento y lenguaje* (Vol. 28). Universitat de València. Retrieved from https://books.google.com/books?id=hiaqpw7WsXkC&pgis=1

Manaris, B. Z., & Slator, B. M. (1996). *Interactive Natural Language Processing: Building on Success*. Computer, IEEE.

Martí, M. A., & Taulé, M. (2011). La Academia y la investigación universitaria en las tecnologías de la lengua. Retrieved March 18, 2015, from https://docs.google.com/file/d/0B6N0v65RwfFSN1RBWGtWVmpLTXc/edit?pli=1

N, S., Haro, G., & Gelbukh, A. (2007). *Investigaciones en análisis sintáctico para el español*. Instituto Politécnico Nacional. Dirección de Publicaciones. Retrieved from http://www.gelbukh.com/libro-investigaciones/LibroSint.pdf

Oatley, K. (1992). *Best Laid Schemes: The Psychology of the Emotions*. Cambridge University Press. Retrieved from https://books.google.com/books?id=H14npd9i_icC&pgis=1

The Eagles Lexicon Interest Group. (2011). ETIQUETAS EAGLES. Retrieved April 13, 2015, from http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html

Vilares Ferro, J. (2005). *Aplicaciones del procesamiento del lenguaje natural en la recuperación de información en español*. Universidade Da Coruña. Retrieved from http://coleweb.dc.fi.udc.es/cole/library/ps/Vil2005a.pdf