Adaptación, optimización y expansión de Ecode un sistema extractor de contextos definitorios

Tesis para obtener el título de ingeniero en computación presenta:

José Luis Vieyra Sagaón

Asesor Dr. Gerardo Sierra Martínez

Índice de figuras

Índice de tablas

A.·. L.·. G.·. A.·. D.·. U.·.

A mi madre

Te agradezco por el hombre que me hiciste
ya que todo lo que soy lo soy por ti.

Agradecimientos

Primero y antes que nada dedico este trabajo íntegramente a la memoria de mi madre, que por obra del destino no pudo estar presente para ver este trabajo realizado y es gracias a ella que me he convertido en Ingeniero.

Además quiero agradecer a mis sinodales, en especial a Gerardo Sierra por el tiempo que me ha permitido permanecer en el GIL y el camino académico que me ha mostrado y apoyado a seguir; a Alfonso Medina por todo su apoyo en tiempos de crisis, su cariño y los conocimientos que me ha permitido desarrollar.

A Flor, por tu gran apoyo en los momentos más difíciles, por tu amor incondicional, tu risa que me pone de buenas, tus besos tiernos llenos de amor, por todos mis sueños que has hecho realidad, por cada risa, cada minuto, cada caricia y por todo lo que me haces sentir.

A mis amigos tan queridos del grupo de Ingeniería lingüística, Irasema mi queridísima sentida hermana, a Josh por tantas risas y una gran amistad, a Victor por su gran apoyo en la realización de éste trabajo, por su amistad incondicional y por viajes increíbles que hemos vivido, y la protección que nos brinda con sus avanzados conocimientos de capoeira, a Alejandro por su siempre amena plática y su amistad, a Teresita por aguantar tanta carrilla, por sus siempre rápidas correcciones y por el gran cariño que nos tenemos, a Pavel por ser un gran amigo al que aprecio mucho y por las tantas experiencias que hemos vivido juntos (aunque guste de continentes enteros), a Azuri, Tavito, Jessy, Brenda, en fin a todo el grupo.

A todos mis queridos hermanos que siempre me apoyan a seguir adelante con fuerza y estabilidad, por permitirme conocer tantas cosas que no podemos ver a simple vista.

A todos mis familiares en especial a mi hermana Gabriela, que me apoyaron en tan complicada situación por la que pasé, a mi tía Elly, Erika, David, tía Chata, Teacher y a mis primos en especial a mi ahijado Santiago que me da fuerzas para continuar luchando.

A todos mis amigos por su gran amistad y tan buenos momentos, Benjamin “el amarrado”, a Cabañero, Luis, Memo, Bal, Gaby, Brenda, Andres, el gordo, Cristobal, Mariana, Atte y todos aquellos que no están aquí nombrados pero presentes en mi corazón

A todos mis profesores de la carrera que con el granito de arena de cada uno he podido construirme como un Ingeniero, Orlando Zaldivar, Oscar Valdez, Laura Sandoval, Paquito, Iriarte, en fin a todos ellos.

Termino agradeciendo al universo entero por permitirme conocer a todos ustedes y le pido que me permita seguir contando con su amistad y cariño.

A todos ustedes y los que no están aquí:

¡Gracias!

1. Introducción

1.1. Antecedentes

Este trabajo surge dentro del marco del proyecto *Extracción terminológica en textos de especialidad*, llevado a cabo dentro del Grupo de Ingeniería Lingüística (GIL) del Instituto de Ingeniería de la UNAM, con el apoyo del Consejo Nacional para la Ciencia y la Tecnología (CONACyT), proyecto 82050.

Este proyecto, y en particular esta investigación, se ha enfocado en la extracción automática de contextos definitorios especializados. Particularmente, se ha creado un sistema de extracción automática llamado Ecode (Extractor de Contextos Definitorios) (Alarcón, 2009).

Con anterioridad, dentro del mismo GIL, ya se han realizado trabajos y tesis dentro de este mismo proyecto orientados a la extracción de contextos definitorios en textos especializados de manera automática. .Con respecto a investigaciones teóricas sobre los contextos definitorios se han desarrollado trabajos tales como: *Análisis Lingüístico de definiciones en contextos definitorios* (Aguilar, 2009), *Análisis lingüístico de definiciones analíticas para la búsqueda de reglas que permitan su delimitación automática* (Hernández, 2009) y *La funcionalidad al interior de contextos definitorios con definiciones analíticas: el patrón sintáctico* para + *infinitivo* (Sánchez, 2009). Asimismo, se han publicado trabajos prácticos como: *Agrupamiento semántico de contextos definitorios* (Molina, 2009) o *Metodología de elaboración para un corpus informático de contextos definitorios* (Mijangos, 2011).

Los trabajos del Dr. Rodrigo Alarcón con relación al desarrollo del Ecode han sido: *Análisis lingüístico de contextos definitorios en textos de especialidad* (Alarcón, 2003) y *Descripción y evaluación de un sistema basado en reglas para la extracción automática de contextos definitorios* (Alarcón, 2009).

1.2. Problemática

El problema que aborda esta tesis surge a raíz de los trabajos realizados con anterioridad, especialmente de los elaborados por Rodrigo Alarcón orientados a la extracción automática

de contextos definitorios. Como ya se ha dicho, este proceso se ha estado llevando a cabo por una herramienta denominada Ecode. Sin embargo, los trabajos de Rodrigo Alarcón aún requerían de refinamiento y necesitaban ser abordados desde una perspectiva desde el punto de vista ingenieril.

Es necesario que el Ecode extraiga de una manera más eficiente y rápida los contextos definitorios, y que la precisión y la exhaustividad de tal herramienta mejoren. Se busca que el Ecode sea una herramienta flexible y adaptable, es decir, que opere conforme a las necesidades que los usuarios (investigadores, alumnos, becarios, etc.) necesiten.

1.3. Objetivos de la tesis

Con relación a esta problemática y a lo dicho con anterioridad, me plantee los siguientes objetivos para la realización de esta tesis enfocados a las necesidades expuestas.

1.3.1 Objetivos generales

Desarrollar una herramienta integral que no requiera procesos intermedios para la obtención de resultados, integrando plenamente los procesos internos y los de entrada y salida que son requeridos.

1.3. 2 Objetivos particulares

- Desarrollar para Ecode un módulo de entrada que flexibilice la alimentación del texto requerido y lleve a cabo todo el pre-procesamiento requerido para que el sistema funcione.
- Optimizar el sistema proponiendo acciones concretas de ajuste al algoritmo y al código.
- Definir un módulo de salida que permita entregar resultados fáciles de interpretar y de integrar a otros sistemas.
- Generar una interfaz de usuario.
- Reescribir porciones del código redundantes o improductivas.

1.4. Estructura de la tesis

Con respecto a las necesidades y objetivos de esta investigación, se plantea comenzar la tesis por los temas más generales para abarcar al final los más particulares y mostrar las conclusiones a las que se llegó. La estructura de la tesis es la que se muestra a continuación:

1. Conceptos básicos: En este apartado planteo los conceptos que utilizaré a lo largo de la tesis. De manera general, se explica en qué consiste el procesamiento de lenguaje natural (PLN), la extracción terminológica, los contextos definitorios, el etiquetado POS, la lematización y los corpus lingüísticos. Por otra parte, también explicaré las herramientas utilizadas en el sistema final.
2. Ecode, una herramienta de extracción de CDs en textos de especialidad: Este capítulo aborda la temática acerca del sistema Ecode, su descripción, el algoritmo utilizado, las entradas y salidas del sistema, etc. También se analiza la funcionalidad del Ecode y del Describe, herramienta para la cual fue creado el Ecode.
3. Desarrollo y trabajo sobre Ecode: Este es el capítulo central para la tesis, pues aquí se describe el trabajo que se realizó sobre el primer Ecode, su optimización, la implementación web que se hizo de este y en general se describirá el trabajo que realicé conforme a las necesidades y objetivos planteados en el marco de la investigación de esta tesis.
4. Resultados y pruebas: En este apartado se muestran las pruebas que se realizaron con las optimizaciones realizadas al sistema y los resultados que se arrojaron. Se comparan los resultados estadísticos de precisión y exhaustividad conforme a los del Ecode original y se medita sobre los objetivos planteados en la introducción con base en los resultados obtenidos.
5. Conclusiones: Finalmente, se muestran las conclusiones sobre los logros alcanzados en esta tesis, su utilidad y el trabajo a futuro.

2. Conceptos básicos

En este capítulo se describen brevemente los conceptos básicos divididos en dos secciones: por una parte, el procesamiento del lenguaje natural (PLN) en donde se detallan las nociones propias de esta rama de la computación; y, por otra parte, las herramientas computacionales utilizadas.

2.1. Procesamiento de lenguaje natural

La inmensa cantidad de información que se maneja diariamente en el ámbito académico, empresarial y de gobierno requiere que sea procesada debidamente para ahorrar tiempo y recursos en su interpretación y análisis. Por tanto, la ingeniería en computación, en especial la inteligencia artificial, abordan este problema a través del procesamiento del lenguaje natural (PLN).

El PLN es una rama de las ciencias de la computación, relacionada con la inteligencia artificial y la lingüística. El PLN elabora sistemas computacionales para la comunicación eficiente entre personas y máquinas a través de lenguajes naturales. Diseña mecanismos para que los programas ejecuten o simulen la comunicación. Implica aspectos cognitivos, de memoria y de comprensión del lenguaje.

Entre las aplicaciones del PLN destacan la minería de textos y la recuperación de información. La minería de textos se refiere a un conjunto de técnicas estadísticas o lingüísticas, que permiten la extracción en textos de información de manera automática. La recuperación de información (RI) “es la elaboración de sistemas para la búsqueda y selección de documentos que cumplan ciertos criterios señalados por un usuario” (Alarcón, 2009: 30). La extracción de información (EI) “se encarga de desarrollar sistemas para la búsqueda y selección de datos específicos sobre eventos, entidades o relaciones a partir de un conjunto de documentos”. (Alarcón, 2009: 30).

Sosa (1997) declara que “el PLN se concibe como el reconocimiento y utilización de la información expresada en lenguaje humano a través del uso de sistemas informáticos”.

2.1.1. Extracción terminológica

La extracción terminológica trata sobre la identificación de términos. Para esto, utiliza la terminología. Según Cabré (1993: 82):

> La terminología puede ser entendida, por un lado, como "el conjunto de principios y de bases conceptuales que rigen el estudio de los términos", y por otro, como "una materia de intersección que se ocupa de la designación de los conceptos de las lenguas de especialidad", cuyo objetivo, a grandes rasgos, es la denominación de los conceptos.

Sager (1993: 35), por su parte, define estas tres concepciones de la terminología de la siguiente forma:

1. El conjunto de prácticas y métodos utilizados en la recopilación, descripción y presentación de términos;
2. una teoría, es decir, el conjunto de premisas, argumentos y conclusiones necesarias para la explicación de las relaciones entre los conceptos y los términos;
3. un vocabulario de un campo temático especializado.

La terminología surgió gradualmente como una disciplina lingüística separada cuando personas como Wüster opinaron que los términos deberían ser tratados de manera diferente que las palabras del lenguaje general (Pearson, 1998):

1) Porque en contraste con la lexicología, donde la unidad léxica es el punto de inicio, el trabajo de la terminología empieza desde el concepto. Un concepto consiste en un agregado de características que pueden ser reconocidas como comunes a un número de objetos individuales y que pueden usarse como medios para el ordenamiento mental y la comunicación. Un concepto es un elemento del pensamiento.

2) Porque los terminólogos se interesan en el vocabulario aislado. Los términos son diferentes de las palabras no solo en cuanto a su significado sino a su naturaleza y uso. Hay una correspondencia entre el término como una etiqueta y el concepto como un constructo mental, e idealmente, un término se refiere únicamente a uno y solo un concepto de un campo determinado.

3) Los terminólogos se ocupan de imponer normas para el uso del lenguaje. Esto llevó a la creación de vocabularios estandarizados que se usarían como un medio para representar las estructuras conceptuales que llevan cada campo de conocimiento.

El dominio de una materia especializada contiene una serie de conceptos o constructos mentales que están representados por términos. La relación entre conceptos y términos está estandarizada. La relación entre conceptos está representada por relaciones lógicas y ontológicas que se utilizan para construir sistemas jerárquicos de conceptos (Pearson, 1998).

La EI elabora sistemas para la extracción automática de términos. La EI se vale de sistemas basados en reglas lingüísticas o en reglas estadísticas para el análisis de textos que permitan obtener candidatos a términos. También se ha dedicado a la obtención de contextos definitorios para inferir el significado de los términos de acuerdo con sus características, sus atributos o sus relaciones con otros términos. Así, se convierte en un proceso enfocado a extraer relaciones semánticas (RSs) y contextos definitorios (CDs).

2.1.2. Contextos definitorios

"Un contexto definitorio es un fragmento textual en donde se obtiene información para conocer el significado de un término. Son unidades discursivas que vehiculan información predicativa sobre un término en un dominio de conocimiento específico." (Alarcón, 2009: 39)

Los contextos definitorios no solo aportan información sobre el significado del término sino que también permiten conocer las relaciones semánticas que existen con otros términos y de esta manera establecer una red conceptual del campo del conocimiento al que pertenecen.

De Bessé (1991): entiende por contexto al punto de inicio de cualquier trabajo terminográfico. El contexto es el entorno lingüístico de un término conformado por un enunciado, es decir, las palabras o frases alrededor de dicho término, y que persigue dos funciones básicas: aclarar el significado de un término e ilustrar su funcionamiento.

Por tanto, los contextos constituyen un elemento esencial para la descripción de un concepto y resultan indispensables para redactar una definición.

De Bessé distingue los CDs como aquellos contextos donde se aporta información sobre los atributos de los términos. Diferencia los contextos conceptuales como aquellos que se refieren a características sobre las relaciones conceptuales de los términos, en tanto los materiales proveen instrucciones sobre el alcance de los términos y la forma en que éstos operan en un contexto determinado.

La importancia de los contextos definitorios y las relaciones semánticas en las prácticas terminográficas radica en la necesidad de comprender un término y situarlo frente a otros de su campo, a través del análisis de las situaciones reales en las que estos se definen dentro de la comunicación del lenguaje especializado.

Un contexto definitorio (CD) es un fragmento textual que, mediante patrones verbales, patrones pragmáticos, tipografía y nexos, engloba un término y su definición la cual puede ser extraída automáticamente.

De Alarcón, Bach y Sierra (2008:2).

> Se entenderá por contexto definitorio a todo aquel fragmento textual de un documento especializado donde se define un término. Estas unidades están formadas por un término (T) y una definición (D), los cuales se encuentran conectados mediante un patrón definitorio (PD), por ejemplo verbos como definir o entender. Opcionalmente, un contexto definitorio (CD) puede incluir un patrón pragmático (PP), esto es, estructuras que aportan condiciones de uso del término o que matizan su significado, por ejemplo en términos generales o en esta investigación.

En la figura 2.1 se ve como un CD, contiene un término (T), un patrón definitorio (PD) y una definición (D), adicionalmente puede o no contener un patrón pragmático (PP).

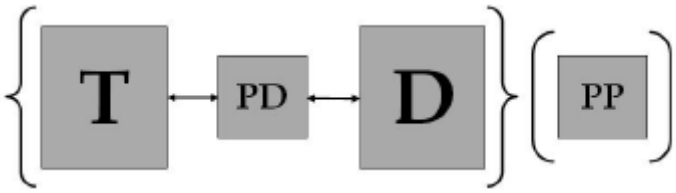

Figura 2.1. Estructura de un CD (Sierra, 2009)

Existen dos tipos de patrones definitorios: los patrones tipográficos, y los patrones sintácticos.

Sierra (2009:18) comenta sobre los patrones tipográficos:

> La tipografía de un texto es un recurso que sirve como ayuda visual para identificar fácilmente los elementos importantes y diferenciarlos del resto del texto común. En muchos casos, los términos tienden a ser frecuentemente resaltados. Muchas veces ocurre que la definición también se encuentra señalizada con algún elemento tipográfico o con alguna tipografía específica. En este sentido, los patrones tipográficos se utilizan ya sea para resaltar a los elementos constitutivos mínimos de los CDs o bien para conectar dichos elementos.

Y sobre los patrones sintácticos (Sierra, 2009:19):

> Un camino para extraer CDs de manera automática en textos de especialidad consiste en identificar las estructuras sintácticas recurrentes tanto de los elementos mínimos constitutivos como de los conectores que unen a estos dos elementos. Alarcón (2009) describe dos patrones sintácticos que sirven para conectar el término con su definición.

> Cuando dichos conectores tienen como núcleo un verbo, tenemos entonces un patrón verbal definitorio (PVD). Cuando se emplean otro tipo de formas sintácticas cuya finalidad es establecer una reformulación de una idea o concepto, y que se utilizan para esclarecer el significado de un término, tenemos marcadores reformulativos.

Una clasificación de los contextos definitorios es por el tipo de su definición y pueden ser (Alarcón, 2009: 135-137):

- *Analíticos.* En los cuales se presenta tanto el *género próximo* al cual pertenece el término, como la *diferencia específica* que lo diferencia de otros elementos de su campo de conocimiento; un ejemplo extraído del corpus de Alarcón (2009):

  > *De acuerdo con el enfoque integral expuesto, <t>el sistema de gestión</t> se concibe como <d><genus>una organización,</genus> <differentia>cuyo funcionamiento busca lograr ciertos objetivos a través de la operación de los diversos subsistemas interrelacionados que la componen</differentia></d>.*

- *Funcionales.* En estos contextos no se incluye el género próximo pero se indica la función del término. P. ej. de Alarcón, (2009):

  <t>Las escalas de Likert</t> se usan habitualmente <d>para <función>cuantificar actitudes y conductas</función></d>.

- *Extensionales.* Como en el caso anterior, en estos contextos tampoco se presenta el género próximo, sin embargo se enumeran características que conforman las partes de la entidad. P. Ej de Alarcón, (2009).

  > *El terminal utilizado es <t>el videoteléfono</t>, que consta básicamente de <d><partes>una pantalla, cámara, teclado, micrófono, altavoz</partes></d>.*

- *Sinonímicos.* En este tipo de contextos se presenta otro término que puede equivaler semánticamente al término definido. P. ej. de Alarcón (2009):

En la mujer, <t>el conducto vaginal<t> se llama también <d><termino equivalente>conducto de Nuck</termino equivalente></d>.

Recapitulando, las partes constitutivas de un CD son el término y su definición; ambos elementos se encuentran conectados por un patrón definitorio (PD) el cual puede ser tipográfico: por medio de marcadores discursivos como son que el texto esté en negritas o subrayado o que este precedido por dos puntos, y algunos otros símbolos (Patrón tipográfico definitorio); o sintáctico: por medio de verbos definitorios y sus nexos (Patrón Verbal Definitorio). Pudiendo contener patrones pragmáticos en su estructura.

Cabré (1993) define a los patrones pragmáticos como un tipo de información relevante para situar al término dentro del contexto en el cual aparece. Esta información describe el uso de los términos y precisa las condiciones de uso o de alcance de dicho término, como son el ámbito temático, la ubicación geográfica, las instituciones que utilizan el término, el nivel de especialidad, o la frecuencia de uso, entre otras características pragmáticas.

Un ejemplo de contexto definitorio obtenido de es:

<t> Las escalas de Likert </t><PVD>se usan</PVD> <PP>habitualmente</PP> <Nx>para</Nx><d> cuantificar actitudes y conductas.</d> (Alarcón, 2009: 136).

2.1.3 Etiquetado de partes de la oración

Las partes de la oración (también conocidas como POS –de sus siglas en inglés, *part of speech*-, clases de palabras, clases morfológicas, categorías gramaticales y etiquetas léxicas) son muy importantes para el PLN por la información que proporcionan de una palabra y las palabras adyacentes.

Las partes de la oración se utilizan para la recuperación de información (RI) y para extraer verbos o sustantivos de diferentes textos. La localización automática de las partes de la oración es útil en el análisis sintáctico, en algoritmos de desambiguación, en análisis sintácticos superficiales de palabras, para encontrar nombres, datos, fechas en textos y para otras entidades de la extracción de información. Finalmente, los corpus que han sido marcados con partes de la oración se usan para la investigación de la lengua.

Sierra y Alarcón (2010:2) definen al etiquetado de partes de la oración de la siguiente manera:

> El siguiente nivel en PLN que sirve para la búsqueda de información es el análisis de las partes de la oración y su etiquetado.
>
> Este proceso consiste en identificar la categoría gramatical (ya sea verbo, adjetivo, sustantivo, etc.) de cada palabra y asignarle una etiqueta que será utilizada posteriormente en los sistemas inteligentes de extracción de información. Este proceso sirve además para otros niveles superiores de análisis en PLN. Por ejemplo, si queremos encontrar imágenes que hagan referencia al instrumento musical bajo, es probable que también se recuperen objetos cuya característica implique la palabra bajo como adjetivo o preposición.

Para Méndez (2009:72):

> El problema de la identificación automática de categorías gramaticales puede entenderse como el problema de asignar cierta etiqueta, que corresponde a una categoría, a cada palabra de un corpus (cf. Charniak 1996/1993)[1]. Para Voutilainen (1999a: 3)[2] las etiquetas son símbolos descriptivos que se asignan a palabras en un texto ya sea de forma manual o automática. Entonces estas etiquetas codificarían el nombre de la categoría y los rasgos morfosintácticos asociados.

A las etiquetas que se utilizan para marcar las palabras se les llama etiquetas de partes de la oración o etiquetas morfosintácticas. Cuando en un corpus se decide con qué etiquetas se va a etiquetar, se tiene un conjunto de etiquetas de corpus. Existen dos consideraciones fundamentales para dichas etiquetas: la especificidad de las categorías, que es el nivel de detalle con el que será etiquetados y por otro lado las etiquetas en sí y cómo se codifican dichos rasgos en ellas (Méndez, 2009). En la tabla siguiente se muestra el conjunto de etiquetas del Corpus del Español Mexicano Contemporáneo (CEMC):

[1] Charniak, Eugene. 1996/1993. *Statistical language learning*. Cambridge, Massachusetts: The MIT Press.

[2] Voutilainen, Atro. 1999ª. "*Orientación*", en Hans Halteren (ed.) Syntactic Wordclass Tagging, Dordrecht. Netherlands: Klluwer Academic, 3-7.

Etiqueta	Categoría
0	Ambigua
1	Adverbio
2	Adjetivo
3	Conjunción
4	Preposición
5	Pronombre
6	Artículo
7	Contracción
8	Nominal
9	Verbo
A	Apoyos conversacionales
B	Nombres propios
C	Otros (cifras, errores y palabras que comenzaban con mayúscula)

Tabla 2.1 Conjunto de etiquetas del CEMC Méndez (2009: 75)

Existe un estándar para las etiquetas gramaticales que es multilingüe y vale la pena mencionar: el estándar EAGLES (Expert Advice Group for Language Engineering Standars). Se ha trabajado en una serie de lineamientos para el etiquetamiento de POS de diversas lenguas (danés, alemán, inglés, francés, holandés, griego, italiano, portugués y español) y la posibilidad de adaptarse a las particulares de alguna lengua en específico (Leech & Wilson 99).

La codificación EAGLES se basa en un conjunto de letras para las categorías (atributos obligatorios), rasgos morfosintácticos (atributos recomendados) y particularidades de cada lengua (atributos opcionales), ordenadas en posiciones consecutivas.

Para el etiquetado con partes de la oración se utilizan varios algoritmos, desde las reglas elaboradas a mano, los métodos probabilísticos (etiquetado HMM y etiquetado de máxima entropía), hasta el etiquetado basado en la transformación y en la memoria para la etiquetación de partes de la oración (Jurafsky, 2007).

De los muchos algoritmos que existen dos se han aplicado a este trabajo: el algoritmo de etiquetado de Brill y el TreeTagger.

2.1.3.1. Algoritmo de Brill

El modelo Brill es un etiquetador basado en reglas que se desempeña con igual eficiencia que los modelos probabilísticos, obviando las limitaciones comunes de los enfoques basados en reglas. Las reglas se asumen automáticamente. Se requiere menos información almacenada, menos reglas, y se facilita encontrar e implementar mejoras al etiquetado y al traslado de un conjunto de etiquetas o género de corpus a otro. El algoritmo funciona encontrando automáticamente las reglas de etiquetado y reparando su debilidad, por lo que mejora su rendimiento.

El sistema requiere de un corpus previamente etiquetado para entrenar al algoritmo. Se utiliza el 90% para el proceso descrito a continuación, 5% para la selección de los parches (reglas de etiquetado manuales) que arrojan menos resultados y 5% para pruebas.

El etiquetador inicia asignando al corpus de trabajo la etiqueta POS más probable de acuerdo con el corpus de entrenamiento, sin considerar información contextual; toma en cuenta las palabras que inician con mayúscula como nombres propios. Las palabras que le son desconocidas de acuerdo con el corpus de entrenamiento las etiqueta de acuerdo con las 3 últimas letras de las mismas.

El etiquetado anterior arroja resultados cercanos al 90% (Brill 1992: 113), posteriormente se procede al análisis de los errores etiquetando el 5% del corpus de entrenamiento (corpus de parches) con las categorías encontradas y comparándolas con las que ya fueron asignadas por un experto. Con los errores encontrados se procede a aplicar una serie de reglas (parches) previamente elaboradas que alimentan al sistema.

Se aplican dichos parches y se compara cuántos errores se corrigieron; el sistema repite este proceso hasta encontrar cuáles son las reglas que eliminan la mayor cantidad de errores y las guarda.

Posteriormente, cuando se inserta al sistema un texto no etiquetado, se procede etiquetando primero con las categorías gramaticales más probables y después aplicando cada parche, en orden, hasta producir la salida final.

El etiquetado tiene la particularidad de elegir solamente las mejores reglas para el etiquetado, por lo que permite ingresar una gran cantidad de reglas sin que se afecte el resultado final, lo que facilita la elección de las reglas más sencillas y más efectivas para el etiquetador.

El etiquetador Brill es portátil. Es transferible a otros conjuntos de etiquetas o géneros y a otros idiomas. Si el etiquetador se utilizara en un corpus diferente se encontraría un conjunto distinto de parches adecuado.

En este etiquetador basado en reglas, la información se captura con menos de ocho reglas, facilitado el desarrollo posterior del etiquetador. La información contextual se expresa de manera compacta (Brill, 1992).

2.1.3.2. TreeTagger

TreeTagger es un etiquetador POS probabilístico, robusto y portable, trabaja de manera similar a un etiquetador de trigramas con algunas diferencias que explicaré a continuación. Para funcionar requiere de un lexicón de palabras completas, un lexicón en forma de árbol de sufijos y una categoría default.

Del corpus de entrenamiento etiquetado se calculan las probabilidades de las categorías de acuerdo a los trigramas, para esto se normaliza procurando que a los trigramas raros o mal formados se les asigne un valor diferente de cero, pero muy bajo y se normalizan las probabilidades.

El etiquetador estima la transición de probabilidades generando un árbol binario de decisión a partir del corpus de entrenamiento. El árbol de decisión automáticamente determina el tamaño apropiado del contexto que se utiliza para definir las probabilidades de transición.

Un trigrama se determina siguiendo la senda marcada por el árbol hasta que se alcanza la última hoja. Cada hoja contiene un vector de probabilidades de categorías. Posteriormente, se recorta el árbol de trigramas siguiendo una medida de cantidad de información, para eliminar hojas y nodos con poco contenido de información. Como otros etiquetadores probabilísticos, TreeTagger determina la mejor etiqueta utilizando el algoritmo de Viterbi (Forney, 1973).

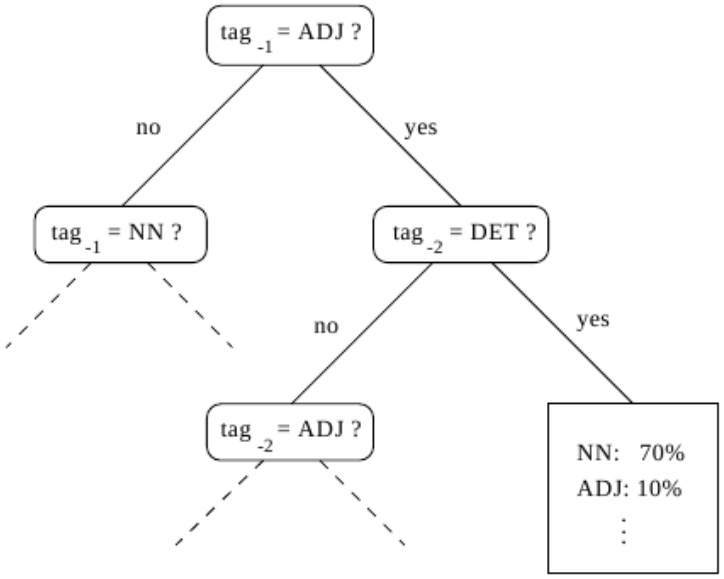

Figura 2.2 Un árbol de decisión de TreeTagger (Schmid, 1994:3)

TreeTagger requiere, a priori, las probabilidades de las categorías y para esto se apoya de dos lexicones que son, el primero, una lista de palabras completas con sus respectivos vectores de probabilidad, el segundo, un árbol de sufijos en cuyos nodos existen caracteres y en sus hojas los vectores de probabilidad de las categorías. Además, con un análisis de los lexicones se obtiene un vector default de probabilidades que permite desambiguar palabras desconocidas.

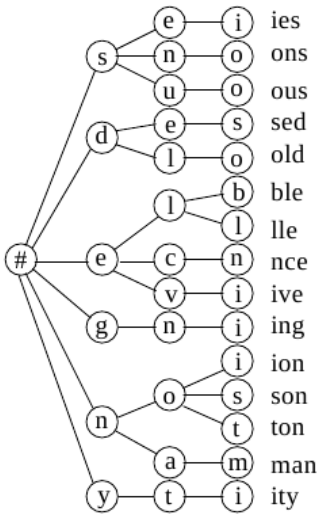

Figura 2.3 Árbol de sufijos de longitud tres (Schmidt, 1994:5)

El algoritmo difiere de otros métodos probabilísticos en la forma que las probabilidades de transición son estimadas, específicamente con un árbol de decisión. TreeTagger es robusto respecto al corpus de entrenamiento y no pierde precisión con un corpus de entrenamiento pequeño como en otros etiquetadores por trigramas. TreeTagger corta el tamaño del contexto donde es necesario para obtener estimados de probabilidad confiables (Schmidt, 1994).

2.1.4 Lematización

La lematización se puede entender como el proceso informático de reducir una palabra a su forma más simple, llamada lema. A veces se confunde el término lematización en español con *stemming,* el cual consiste en truncar los morfemas funcionales de una palabra. Para esto Jurafsky (2006:53) define:

> We will introduce some related algorithms in this chapter. In some applications we don't need to parse a word, but we do need to map from the word to its root or stem. For example in information retrieval and web search (IR), we might want to map from foxes to fox; but might not need to also know that foxes is plural. Just stripping off such word endings is called stemming in IR. We will describe a simple stemming algorithm called the Porter stemmer.
>
> For other speech and language processing tasks, we need to know that two words have a similar root, despite their surface differences. For example the words sang, sung, and sings are all forms of the verb sing. The word sing is sometimes called the common lemma of these words, and mapping from all of these to sing is called lemmatization.

Además Srivastava (2009:15) da una definición más sencilla:

> A simple example of semantic similarity mapping is stemming, that consists of removing inflection from words.

Como bien menciona Jurafsky, el algortimo de Porter es el algoritmo más conocido de lematización. De palabras del propio Porter (1980:1), que trabaja removiendo sufijos:

> Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by removal of the various suffixes -ED, -ING, -ION, IONS to leave the single term CONNECT. In addition, the suffix stripping process will reduce the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is always advantageous…

> …Assuming that one is not making use of a stem dictionary, and that the purpose of the task is to improve IR performance, the suffix stripping program will usually be given an explicit list of suffixes, and, with each suffix, the criterion under which it may be removed from a word to leave a valid stem.

El algoritmo de Porter trabaja con una serie de reglas para eliminar los sufijos de las palabras. Primero se calcula el número de consonantes y vocales en cada palabra y se le asigna un número. Después se aplican las reglas escritas de acuerdo al tamaño de la palabra y la sustitución de dicha regla. Un ejemplo de una regla es:

Tamaño	Sufijo	Reemplazo de sufijo	Ejemplo
m>0	tional	Tion	conditional > condition

Tabla 2.2 Ejemplo de regla en el algoritmo de Porter (Porter 1980:2).

2.1.5 Corpus

Un corpus se define en su forma más simple como cualquier recopilación de muestras de lenguaje. Torruella y Llisterri (1999) entienden que, para la filología, un corpus es una recopilación de textos bien organizada. Cabe decir que, actualmente, los corpus cuentan con soporte informatizado y herramientas computacionales que ayudan en su procesamiento. Para Sierra (2008:446) un corpus es:

> Un conjunto de datos reales y aceptables, debidamente ordenado, codificado y organizado, de diferentes textos recopilados, pertenecientes a un código lingüístico determinado, oral o escrito.

Comúnmente se menciona que un corpus debe contar con las características de variedad y representatividad de la muestra de lenguaje que estén representando (Sierra, 2008).

2.1.5.1 Corpus Técnico del IULA

El corpus técnico del IULA (Cabré y Vivaldi, 1997) es un corpus creado por el Instituto Universitario del Lingüística Aplicada de la Universidad Pompeu Fabra en Barcelona. Este corpus cuenta con alrededor de 10 millones de palabras en español. Para acceder a este, se utiliza la herramienta de búsqueda *Bwananet* que permite una búsqueda básica, estándar y compleja.

Cuenta con diferentes áreas de especialidad en diferentes idiomas, como español, catalán,

inglés, francés y alemán. A continuación muestro una tabla de la cantidad de palabras por idioma y área de especialidad:

Área	Catalán	Español	Inglés	Francés	Alemán	Total
Derecho	1 463 000	2 085 000	431 000	44 000	16 0000	4 039 000
Economía	1 776 000	1 091 000	274 000	78 000	27 000	3 246 000
Medio ambiente	1 506 000	1 062 000	599 000	230 000	429 000	3 826 000
Informática	655 000	1 277 000	338 000	194 000	83 000	2 497 000
Medicina	2 619 000	4 077 000	1 555 000	27 000	198 000	8 476 000
Total	8 019 000	9 542 000	3 197 000	573 000	753 000	22 084 000

Tabla 2.3. Número de palabras por lengua y ámbito (Cabré y Bach, 2004:174)

Es importante mecionar que este corpus fue utilizado por el Ecode como corpus de entrenamiento y de pruebas. Una característica del corpus es que se encuentra etiquetado con POS.

2.2 Herramientas utilizadas y el lenguaje Perl y sus módulos

Las herramientas que utilicé, además del lenguaje de programación *Perl*, fueron *antiword* y *elinks*. El programa *antiword* se encarga de la extracción de texto a partir de archivos Word; por otro lado, *elinks* es un explorador web de consola que extrae texto de URL, HTML y XML.

2.2.1 Herramientas externas

Si bien todo el sistema está hecho en Perl, se utilizaron programas externos para funciones específicas que resultó mucho más fácil utilizarlas que programar su funcionalidad. Estas son:

2.2.1.1 Antiword

Antiword[3] es un programa libre distribuido bajo la licencia de software libre GPL, el cual tiene la funcionalidad de extraer el texto del formato de archivo utilizado por Microsoft Word.

Microsoft únicamente mantiene los formatos de su suite de oficina para los sistemas Windows y Mac y no son abiertos ni existe un manual con sus características, por lo que en el caso del Ecode fue necesario buscar una herramienta que lo permitiera y Antiword resultó ser adecuado para proveer esta funcionalidad al Ecode para poder extraer contextos definitorios de archivos realizados en Ms Word.

2.2.1.2 Elinks

Elinks[4] es un navegador web de consola para sistemas tipo Unix creado en 2001 a partir del explorador de consola Links.

Tiene la particularidad de que, además de proveer todas las funcionalidades de un navegador, permite el volcado del texto de la página de entrada hacia la salida estándar de los sistemas tipo Unix. También permite extraer el texto de archivos HTML y XML, además de los sitios web que se le indiquen.

Esta función permite que Ecode obtenga el texto ya sea de una dirección de internet remota o de archivos de usuario en formato HTML o XML para su posterior procesamiento.

2.2.2 Perl y el procesamiento de texto

Perl se define como un lenguaje de programación de propósito general orientado tanto a programación estructurada como programación en objetos y enfocado a la manipulación de texto. Perl se describe como un lenguaje eficiente, sencillo y completo.

Perl es un lenguaje de programación de gran utilidad para el procesamiento de lenguaje natural, debido a la gran cantidad de herramientas, tanto propias del lenguaje como módulos de terceros, razón por la que se eligió en un principio para realizar el Ecode y se mantiene utilizando dicho lenguaje durante el presente trabajo.

---

[3] Sitio web de Antiword: http://www.winfield.demon.nl/
[4] Sitio web de Elinks: http://elinks.cz/

2.2.2.1 Módulos y extensiones de Perl utilizados

Durante la realización del sistema se utilizaron diferentes módulos de Perl. Un módulo es un conjunto de funciones que proveen una cierta funcionalidad, en este caso Perl y su capacidad de procesamiento. A continuación describo brevemente los módulos que se utilizaron en el sistema.

2.2.2.1.1 Perl::DBI

Perl::DBI [5] (Data Base Interface) es un módulo que permite al programador un ambiente estándar que admite la comunicación a diversos manejadores de bases de datos.

En este caso en particular utilicé este módulo para comunicar al Ecode con la base de datos de Describe, de esta manera se utiliza Perl:DBI para comunicarse con la base, extraer el texto bruto y almacenar los resultados de acuerdo con los parámetros que Describe requiere.

2.2.2.1.2 Perl:: Lingua::Stem::Es

Perl::Lingua::Stem::Es[6] (Porter_stem_Es.pm). Se encarga de lematizar el texto en español con el algoritmo de Porter. Esto se hace porque en desarrollos futuros se pueden necesitar los contextos definitorios lematizados para hacer algún tipo de procesamiento; por ejemplo, en Molina (2009) se lematizan los contextos definitorios para agruparlos por su similitud.

2.2.2.1.3 Perl::File::Type

Perl::File::Type[7] se encarga de detectar el tipo de algún archivo de acuerdo con la información contenida, ya sea en la cabecera de este o en la estructura de los datos que contiene. Se utiliza en Ecode para detectar el tipo de archivo de entrada y tratarlo según sea el caso.

---

[5] El sitio web de Perl::DBI es: http://dbi.perl.org/
[6] Para Perl::Lingua::Stem::ES: http://search.cpan.org/~jfraire/Lingua-Stem-Es-0.04/
[7] Y para Perl::File::Type es: http://search.cpan.org/~pmison/File-Type-0.22/

Cabe mencionar que dentro de esta biblioteca no estaban bien diferenciados algunos tipos de archivo, en especial de texto plano, lo que ocasionaba problemas al Ecode. Por ello tuve que modificar una pequeña parte de la librería para que entregara los archivos de texto limpios y facilitar su posterior procesamiento.

2.2.2.1.4 Perl::CAM::PDF

El módulo Perl::CAM::PDF[8] sirve para leer y escribir archivos en formato PDF, este módulo se apega estrictamente a las especificaciones de PDF de Adobe, por lo que archivos que no se apeguen pueden no ser leídos correctamente.

En el caso particular del Ecode se utiliza para aceptar archivos de entrada tipo PDF, se les extrae el texto utilizando este módulo y posteriormente se procesan como cualquier otro texto.

2.2.2.1.5 Perl::Text::Iconv

Perl::Text::Iconv[9] es el módulo encargado de proveer una interfaz entre Perl y el programa estándar de UNIX iconv, que tiene la función de convertir entre diferentes tipos de codificaciones de caracteres.

Ecode lo utiliza para tratar de convertir de cualquier codificación en el texto de entrada hacia UTF-8, que es con la que trabaja por defecto.

Cabe mencionar que la detección de codificaciones es un asunto de extrema complejidad y que este módulo no detecta todas las codificaciones ni todas las variantes de ellas, por lo que, si bien hace un trabajo adecuado, no es tan confiable y puede convertir mal las cadenas de caracteres.

[8] El sitio web es: http://search.cpan.org/~cdolan/CAM-PDF-1.55/
[9] Su sitio es: http://search.cpan.org/~mpiotr/Text-Iconv-1.7/

3. ECODE

El sistema Ecode fue diseñado con la intención de extraer contextos definitorios a partir de textos de especialidad procesados y seleccionados.

Alarcón (2009) desarrolló un sistema que procesa textos y entrega un conjunto de contextos definitorios, mismos que son identificados en un texto etiquetado con las categorías gramaticales (POS) de las palabras. Así, Ecode entrega una lista de CDs con etiquetas estilo XML, mismos que son validados cualitativamente en un archivo de texto plano (Sierra y Alarcón, 2010).

En este capítulo se analiza y describe el sistema de Alarcón (2009), se destaca su funcionamiento, se describen los errores y finalmente se proponen ajustes de optimización y adaptación según las necesidades de los usuarios.

3.1. Descripción y análisis de Ecode

Ecode es un sistema desarrollado en Perl que, a grandes rasgos, utiliza gramáticas y expresiones regulares para extraer patrones lingüísticos en un archivo de texto etiquetado con POS, además de que proporciona una lista de CDs etiquetados en un archivo de texto.

Ecode se compone de nueve módulos, tres gramáticas y su respectivo script de ejecución. Estos módulos son los siguientes:

Módulo	Descripción
01_vds.pm	Etiquetado de verbos definitorios en candidatos
02_pvds.pm	Marcado de patrones verbales definitorios (PVDs)
03_td.pm	Tipo de definición de los candidatos
04_filtro.pm	Filtrado de contextos no relevantes
05_arbol.pm	Árbol de decisión para la identificación de término y definición
06_retagging.pm	Limpieza de etiquetas
07_rankingTyD.pm	Evaluación individual de los términos y definiciones
08_rankingGlobal	Evaluación global de los contextos
09_final.pm	Última limpieza y presentación de resultados

Tabla 3.1 Módulos del Ecode

Por otra parte, las gramáticas son:

Gramática	Descripción
01_gramaticaPOS.pm	Gramática de partes de la oración (módulo de Perl).
02_gramPVDs.txt	Gramática de patrones verbales definitorios y sus diferentes tipos y posiciones.
03_gramFiltro.txt	Gramática de reglas de filtrado de contextos no relevantes.

Tabla 3.2 Gramáticas de Ecode

El proceso de identificación de contextos definitorios se conforma de 3 partes que son:

- Identificación de candidatos (etiquetado de verbos definitorios, PVDs y tipo de definición)
- Filtrado e identificación de término y definición
- Evaluación y presentación de resultados

El diagrama del sistema se puede resumir en la siguiente figura:

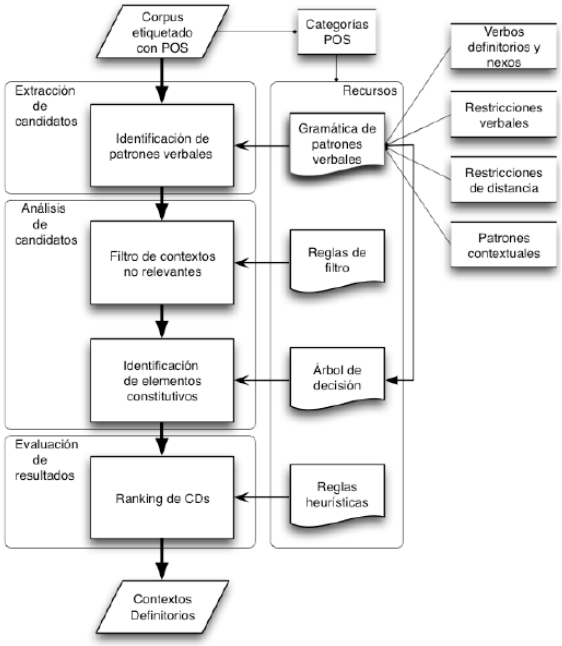

Figura 3.1 Panorama general de la arquitectura de Ecode (Alarcón 2009, 143)

En este punto vale la pena aclarar que el sistema no posee una interfaz de usuario, ni un menú con opciones en consola, sino que simplemente se ejecuta desde la terminal con un archivo de argumento y entrega un segundo archivo de texto con los CDs de salida.

3.1.1. Entrada al sistema

Para iniciar la extracción, el sistema requiere de un archivo de texto separado por líneas en las que cada línea es una oración; una oración se entiende para el fin del sistema como un fragmento textual entre signos de puntuación (como punto, punto y coma o un salto de línea) obtenida de un texto. Cada línea requiere tener una etiqueta que marca el inicio de línea y puede contener información relevante al origen de esta o una numeración.

Por ejemplo:<*doc_codi 00000*>*: La/6 banda/8 de/4 ADN/B que/0 es/9 transcrita/8 se/5 denomina/9 banda/8 codificante/8 ./s* En donde <doc_codi 00000> es una etiqueta de inicio de línea y "La/6" es una palabra etiquetada con su categoría gramatical.

Para el etiquetado de las palabras con su categoría gramatical (POS) se han utilizado diferentes etiquetadores; sin embargo, el conjunto de categorías gramaticales que requiere Ecode no cambia y está basada en el conjunto de etiquetas POS del Corpus del Español Mexicano Contemporáneo (CEMC), aunque originalmente fue pensado para trabajar con el estándar de etiquetado EAGLES. Esto que permite adaptar otros etiquetadores POS al

sistema con una simple sustitución de sus etiquetas. Al iniciar el presente trabajo, el etiquetador POS utilizado era el del CEMC desarrollado por Luis Fernando Lara, basado en el algoritmo de Brill. Veamos las posibles categorías gramaticales que genera este etiquetador:

Codificación numérica	**Categoría**	**Codificación abreviada**
0	Ambigua	Amb
1	Adverbio	Adv
2	Adjetivo	Adj
3	Conjunción	Con
4	Preposición	Pre
5	Pronombre	Pro
6	Artículo	Art
7	Contracción	Ctr
8	Nominal	Nom
9	Verbo	Vbo
A	Apoyos conversacionales	Apc
B	Nombres propios	Npr
C	Otros, como cifras, errores y palabras que comenzaban con mayúscula	Otr

Tabla 2.1[10] Conjunto de marcas del CEMC (Méndez, 2009: 75)

El etiquetador POS, no forma parte del Ecode, por lo que es un proceso externo previo a la ejecución que, para funcionar, requiere las siguientes características:

- La entrada debe tener los signos de puntuación separados de las palabras para su correcto etiquetado. Por ejemplo, "La dopamina no es una proteína; sí lo son, en cambio, las enzimas que sintetizan este neurotransmisor" se debe cambiar por "La dopamina no es una proteína; sí lo son , en cambio , las enzimas que sintetizan este neurotransmisor ."
- Todas las palabras deben estar en minúsculas, de lo contrario pueden ser etiquetadas de manera incorrecta.
- La entrada al etiquetador POS debe codificarse en caracteres iso-8859-1 y es muy

[10] Esta tabla proviene del capítulo 2

sensible a los errores de codificación ocasionando, en gran medida, por un etiquetado incorrecto.

3.1.1.1. Problemas y errores detectados

Los problemas y errores detectados en el proceso son los siguientes:

1. El etiquetador POS es lento, ineficiente e imprime una gran cantidad de texto mientras procesa. Genera archivos intermedios en vez de trabajar en memoria. Además no está integrado de ninguna manera a Ecode.
2. La entrada que recibe es muy específica por lo que los usuarios encuentran dificultad de alimentar el sistema, además de que no tiene un menú, ni interfaz, y sólo recibe texto en bruto, previamente etiquetado con POS, por lo que no es flexible ni accesible. La mayoría del texto se encuentra en archivos de aplicación.
3. ¿Y el punto tres? ¿Se queda vacío?

3.1.1.2. Líneas de trabajo propuestas

Las líneas de trabajo que se propusieron son las siguientes:

1. Desarrollar un módulo de obtención de texto que permita extraer texto en bruto de diferentes fuentes comunes como archivos de aplicación (PDFs, documentos de MSWord, XMLs, HTMLs), de direcciones directamente de Internet (URLs), bases de datos de otros desarrollos (Describe®, Corpus de las Sexualidades en México) y archivos de texto en bruto (txt).
2. Implementar la recepción del texto en bruto e identificar su codificación de caracteres para convertirlo en una codificación adecuada y universal como salida del Ecode, lo más adecuado sería: UTF-8.
3. Crear un separador de oraciones y filtrador de texto basura para evitar la carga del proceso principal y desechar oraciones muy grandes, muy pequeñas o mal estructuradas.
4. Implementar, ya sea un algoritmo o un programa, que etiquete con POS y sea más eficiente, rápido y con mayor precisión en su etiquetado, que trabaje completamente en la memoria y esté plenamente acoplado al sistema.
5. Entrenar y probar dicho etiquetador POS con un corpus de prueba consistente en la posible entrada que los usuarios emplearían para maximizar la precisión del mismo.

3.1.2. Algoritmo General

Al recibir el texto de entrada previamente etiquetado con POS el sistema realiza un proceso principal que consiste en los procesos descritos en la tabla 3.1 y se compone de tres módulos que son:

1. etiquetado de verbos y patrones verbales definitorios y la determinación del tipo de definición
2. filtrado e identificación de los términos y sus definiciones
3. la evaluación y presentación de resultados

A continuación se describe el algoritmo de dichos módulos.

3.1.2.1. Etiquetado de verbos definitorios, patrones verbales definitorios y tipo de definición

(Sierra y Alarcón, 2010) El primer proceso dentro del Ecode es la extracción de candidatos a contextos definitorios:

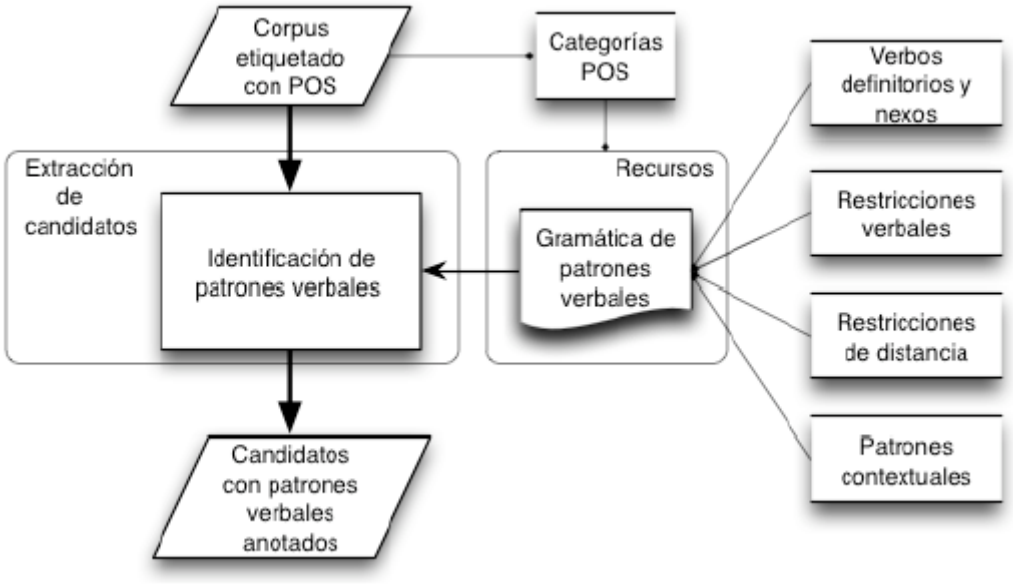

Figura 3.2 Extracción de candidatos a CD (Alarcón 2009: 146)

La entrada es, como ya se mencionó, un archivo de texto con “oraciones” etiquetadas con POS y una etiqueta de inicio de línea. Para la extracción de candidatos se utilizan 2 “gramáticas”: la primera, la gramática de categorías POS (Archivo: 01_gramaticaPOS.pm) que es un módulo de Perl con la definición de las categorías POS y algunos patrones que serán usados durante todo el programa; y la segunda, la gramática de patrones verbales (Archivo: 02_gramPVD.txt) que es un archivo de texto en donde se definen los patrones verbales y las características de ellos, a saber: verbos definitorios que los producen, restricciones a los tiempos verbales, restricciones de distancia entre palabras y patrones conceptuales.

La gramática de categorías POS si bien define reglas de cómo deben estar etiquetadas las palabras, contiene las categorías gramaticales relevantes para el análisis de candidatos y declara variables para uso global en el sistema, es decir, no es una gramática propiamente dicha. En primer lugar, el código de esta contiene estructuras de control y programación orientada a manipular el texto de entrada y no las definiciones de estructuras gramaticales requeridas. Además contiene definiciones de variables que no son categorías POS, pero que sí se requieren a lo largo del programa (por ejemplo, patrones pragmáticos) y también es aprovechada para inicializar arreglos globales.

La gramática de patrones verbales es un archivo de texto con estructura similar a XML que contiene definiciones de los posibles verbos con sus tiempos verbales y demás parámetros para cada tipo de definición de CD. Los parámetros que contiene son:

Parámetro	Posibles Valores	Descripción
Td	Ana, ext, fun, sin	Tipo de Definición
Lm	El infinitivo del verbo	Lema del Verbo Definitorio
Raíz	Regex de la raíz del verbo	Raíz de Verbo Definitorio
raízEx	Regex de la raíz del verbo	Excepción de la Raíz del Verbo Definitorio
Dist	0..n, any	Distancia entre el Verbo Definitorio y su Nexo
Nx	0 | como, por en, etc. ...	Nexo del Patrón Verbal Definitorio
Lt	I=izquierda; N=Nexo; D=Derecha	Lugar del Término dentro del Patrón Verbal Definitorio
lnx	Any	Lugar del Nexo respecto al Verbo Definitorio

Tabla 3.3 Parámetros de la gramática de patrones verbales.

El sistema utiliza estas gramáticas para encontrar los verbos definitorios y este proceso se efectúa en tres pasos, cada uno programado en un módulo de Perl (Archivos: 01_vds.pm, 02_pvds.pm y 03_td.pm):

1. Etiquetado de verbos definitorios (Archivo: 01_vds.pm)

El primer paso es etiquetar las excepciones de verbos definitorios definidos en la gramática de patrones verbales:

Para cada línea en el archivo de entrada

Si la línea contiene una excepción

Se etiqueta la excepción con: <vdEX>excepción</vdEX>

Después el sistema etiqueta los verbos definitorios a partir de las expresiones regulares de las raíces definidas en la gramática:

Para cada línea en el archivo de trabajo

Si la línea empareja con una regex de raíz verbal

Etiqueta el verbo con <vd lemma="lema del verbo">verbo</vd>

2. Etiquetado de los patrones verbales definitorios (PVDs) (Archivo: 02_pvds.pm)

Como ya que se han etiquetado los verbos definitorios, el siguiente paso es etiquetar todos los elementos constitutivos de los patrones, incluyendo los verbos auxiliares, los pronombres, los nexos y las posibles palabras entre el verbo y su nexo:

Para cada patrón verbal definitorio en la gramática

Por cada línea en el arreglo de trabajo que contiene un verbo etiquetado

Si un patrón verbal es encontrado

Etiqueta el nexo con <nx>nexo</nx>

Etiqueta los verbos auxiliares con <aux>verbo auxiliar</aux>

Etiqueta los pronombres con <pr>pronombre</pr>

Etiqueta todo el patrón entre <pdv>patrón verbal</pdv>

3. Etiquetado del tipo de definición, este proceso consiste en etiquetar el tipo de definición de cada candidato considerando la información de la gramática de patrones verbales:

Por cada patrón verbal definitorio en la gramática

Para cada línea del arreglo de trabajo que contiene un patrón etiquetado

Si el lema anotado coincide con el tipo de definición especificado en la gramática

Etiqueta el candidato con <tipoD=”tipo de definición”/>

Un ejemplo de un candidato adecuadamente anotado es:

<tipoD= “analitica”/> El metabolismo<pvd> <pr>se</pr> <aux>puede</aux> <vd lemma= “definir”> definir</vd> <nx>como</nx></pvd> la suma de todos los procesos químicos (y físicos) implicados.

3.1.3. Filtrado e Identificación de término y definición

(Sierra y Alarcón, 2010)Para preparar el análisis de los candidatos se marcan con etiquetas contextuales que sirven como bordes para los procesos automáticos que se llevan a cabo. Todo a la izquierda del patrón verbal es anotado entre <izq /> y todo a la derecha con <der />. Ejemplo:

<tipoD=“analitica”/> <izq>El metabolismo</izq> <pvd><pr>se</pr> <aux>puede</aux> <vd lemma=“definir”>definir</vd> <nx>como</nx><dvp> <der>la suma de todos los procesos químicos (y físicos) implicados.</der>

Después de dicho etiquetado el siguiente paso se divide en 2 procesos: uno, el filtrado de candidatos no relevantes; y el otro, la identificación del término y definición en los CDs.

En el primer caso se aplica un filtro tomando en cuenta que los patrones definitorios pueden ser usados en un rango más amplio de oraciones. En el caso de patrones verbales, algunos verbos tienden a tener un significado metalingüístico más amplio que otros. En el caso de *definir* o *denominar* vs. c*oncebir* o *identificar*, los dos últimos son usados en una variedad de contextos que no necesariamente están expresando información léxica sino…? Además de que algunos verbos con un significado metalingüístico más alto no sólo son usados para definir términos.

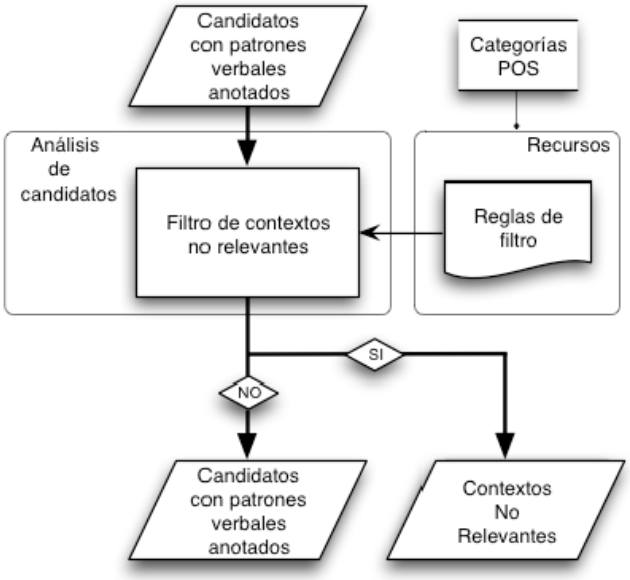

Figura 3.3 Filtro de candidatos no relevantes (Alarcón, 2009: 165)

En este proceso, la entrada consiste en candidatos a CDs con patrones verbales etiquetados. El principal recurso usado para filtrar candidatos no relevantes es un conjunto de reglas de filtrado que usan algunas categorías gramaticales. Cuando los candidatos validan ante alguna de estas reglas, son considerados como candidatos no relevantes. Ejemplos.

Existen partículas o secuencias sintácticas que pueden aparecer cuando los patrones verbales definitorios no son usados para definir un término. Estas partículas y secuencias fueron encontradas en posiciones especificas, por ejemplo: algunas partículas de negación como “no” o “tampoco” que se encontraron en la primera posición antes o después del PVD; los adverbios como “tan”, “poco”, así como secuencias como “poco más”, fueron encontrados entre el verbo definitorio y el nexo “como”, además, se encontraron estructuras o secuencias sintácticas de adjetivo + verbo en la primera posición, después del verbo definitorio.

Posición	Regla
Izquierda	para </izquierda>
Nexo	</dv> .* verbo conjugado .* </nx>
	</dv>.*? se .*?<nx>
	</dv>.*? tanto .*?<nx>
	</dv>.*? sino .*?<nx>
	</dv> , <nx>
	así <nx>
	cerca <nx>
	parte <nx>
	partir <nx>
	más <nx>
	menos <nx>
	mientras (que ,|que) <nx>
	no <nx>
	poco <nx>
	poco más <nx>
	(que ,|que) <nx>
	sino <nx>
	tales <nx>
	tal <nx>
	y <nx>
	ya <nx>
Derecha	<derecha> antes
	<derecha> cuan
	<derecha> para
	<derecha> si
	<derecha> se
	<derecha> verbo conjugado

Tabla 3.4 Reglas de filtrado de contextos no relevantes (Alarcón 2009:167).

El sistema utiliza estos parámetros para filtrar contextos no relevantes. Este proceso se realiza en dos pasos:

1. Etiquetando verbos definitorios. En el que el algoritmo se apareja con todas las instancias de reglas de filtrado encontradas en los candidatos.

Para cada candidato con patrones definitorios verbales
Para cada secuencia en las reglas de filtrado
Si el candidato se apareja con la secuencia
Etiquetar la secuencia con <filtro>secuencia</filtro>

2. Excluir excepciones de la lista de candidatos. Sólo los candidatos no filtrados se dejan después de este paso.

Para cada candidato con patrones definitorios verbales
Si el candidato contiene un filtro
Excluir el candidato de la lista

Un ejemplo de un contexto filtrado como no relevante es el siguiente:

Regla: para </Izq>

<izq>Para</izq><pvd><vd lemma="entender">entender</vd> <nx>como</nx></pvd> <der> funciona el ADN, es necesario conocer algo sobre su estructura y organización</der>.

Una vez que los contextos no relevantes se han filtrado, el siguiente proceso del análisis de candidatos, es la identificación del término y la definición.

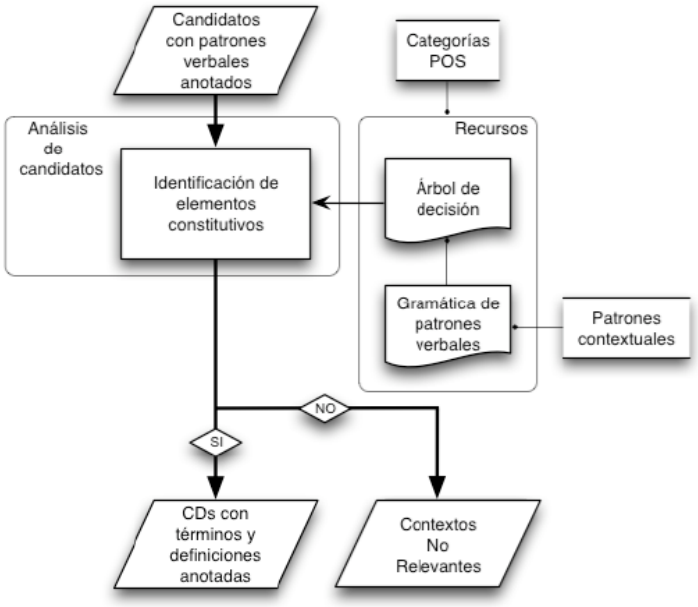

Figura 3.4 Identificación de términos y definiciones (Alarcón 2009: 170)

En este proceso, la entrada consiste en los candidatos a CDs restantes que no han sido filtrados como contextos no relevantes. La identificación del término y la definición se realiza con un árbol de decisión, que toma como fuente principal los patrones contextuales en la gramática de patrones verbales. La entrada también está condicionada ¿como en? el proceso previo, es decir, si los candidatos se aparejan con una de las reglas del árbol de decisión, se consideran como CDs válidos; si no, se filtran como contextos no relevantes. La salida en este caso consiste en CDs con términos y definiciones localizadas.

El uso de patrones contextuales en esta etapa se realiza considerando que, dependiendo de cada PVD, los términos pueden aparecer en algunas posiciones específicas en CDs en español. Algunos verbos definitorios permiten diferentes configuraciones estructurales; entonces, el proceso de identificarlos automáticamente y a sus definiciones está altamente relacionado con la posición de cada elemento constitutivo.

Se utiliza un árbol de decisión para resolver este problema, para detectar por medio de inferencias lógicas las posiciones probables de los términos, definiciones y patrones pragmáticos. Para asimilar esto, se establecieron algunas expresiones regulares simples para representar cada elemento constitutivo:

Elemento	Expresión regular	Descripción
ERT	TRE = BRD (Det) + N + Adj. {0,2} .* BRD	Expreg de Término
ERPP	PPRE = BRD (sign) (Prep | Adv) .* (signo) BRD	Expreg de Patrón Pragmático
ERD	DRE = BRD (Det) + N	Expreg de definición
N	Se obtiene de las etiquetas POS	Nombre
Adj	Se obtiene de las etiquetas POS	Adjetivo
Prep	Se obtiene de las etiquetas POS	Preposición
Adv	Se obtiene de las etiquetas POS	Adverbio
BRD	Es izq o der	Borde

Tabla 3.5 Elementos de la identificación de términos y definiciones

Como en el proceso de filtrado, las etiquetas contextuales funcionan como límites para marcar las instrucciones del árbol de decisión. Además, cada expresión regular podría funcionar como un límite. Una representación de un árbol de decisión se puede apreciar en la siguiente figura:

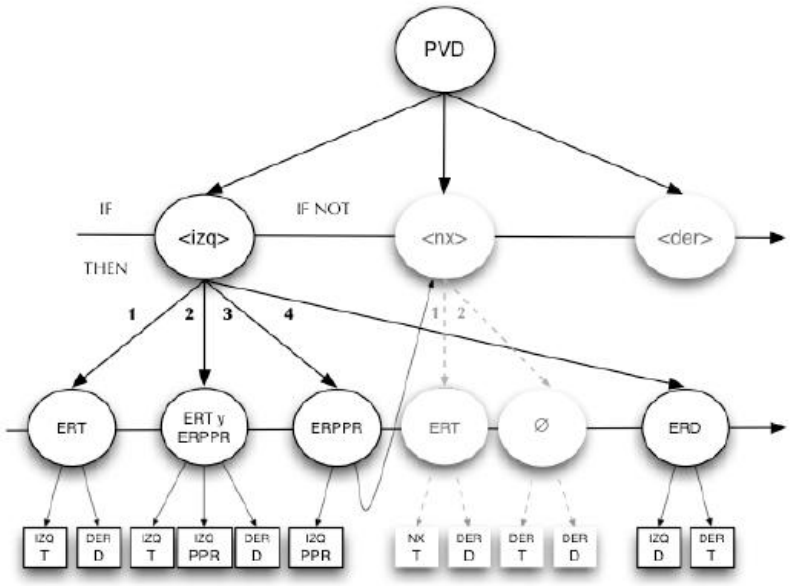

Figura 3.5 Árbol de decisión para la identificación de términos y definiciones (Alarcón, 2009: 175)

En el primer nivel, las ramas del árbol corresponden a posiciones diferentes en las cuales pueden ocurrir (nexo, derecha o izquierda). En el segundo nivel, las ramas corresponden a las expresiones regulares de cada elemento del CD. Los nodos (conjunciones de ramas) corresponden a decisiones tomadas de los atributos de cada rama y también se relacionan horizontalmente con inferencias *SI* o *SI NO*, y verticalmente a través de inferencias *ENTONCES.* Finalmente, las hojas corresponden a la posición asignada de cada elemento constitutivo. Las inferencias del árbol de decisión se explican en la siguiente tabla:

Posición	**Si**		**Entonces**
Nexo		La posición de nexo corresponde solamente a una expresión regular de término:	<nx> = término; <derecha> = definición
		La posición de nexo corresponde a un término y a una expresión regular de patrón pragmático:	<nx> = término; <nx> = patrón pragmático; <derecha> = definición
		La posición de nexo corresponde solamente a una expresión regular de patrón pragmático:	Ir a inferencia 4
Derecha		La posición derecha corresponde a un término y a una expresión regular de definición	<derecha> = término; <derecha> = definición
		La posición derecha corresponde solamente a una expresión regular de término	<derecha> = término; <izquierda> = definición
		La posición derecha corresponde solamente a una expresión regular de definición	Ir a la inferencia 7
Izquierda		La posición izquierda corresponde a un término y a una expresión regular de patrón pragmático.	<izquierda> = término <izquierda> = patrón pragmático; <derecha> = definición
		La posición izquierda corresponde solamente a una expresión regular de término	<izquierda> = término; <derecha> = definición

Tabla 3.6 Inferencias del árbol de decisión (Sierra y Alarcón, 2010: 10)

Para ejemplificar el proceso previo se puede observar los siguientes dos contextos Ejemplo:

> *<izquierda>En sus comienzos</izquierda> <dvp>se <dv lemma="definir"> definió</dv></dvp> <nx>la psicología como</nx> <derecha>"la descripción y la explicación de los estados de conciencia" (Ladd, 1887).</derecha>*

Esto es, el árbol infiere la regla 1: la posición del nexo corresponde solamente a una expresión regular de término. Entonces la posición del nexo corresponde al término (*la psicología*) y la posición derecha corresponde a la definición (*la descripción y la explicación de los estados de conciencia*….). Ejemplo:

> *<izquierda>Una librería genómica</izquierda> <dvp>se <dv lemma="definir"> define</dv></dvp> <nx>tradicionalmente como</nx> <derecha>un conjunto de clones en el que está representado todo el genoma de un organismo.</derecha>*

Aquí el árbol analiza las inferencias 1 a 3 y encuentra la regla 3: la posición de nexo corresponde solamente a una expresión regular de patrón pragmático (*tradicionalmente*); entonces analiza las inferencias 4 a 8 y encuentra la regla 8: la posición izquierda corresponde solamente a una expresión de término regular (*una librería genómica*) y la posición derecha corresponde a la definición (*un conjunto de clones en el que*…).

3.1.3.1. Ranking y presentación de resultados

La evaluación y presentación de resultados se conforma de 4 módulos (Archivos: 06_retagging.pm, 07_rankingTyD.pm y 08_rankingGlobal.pm, 09_final.pm).El sistema incluye un evaluador automático de resultados de los candidatos que fueron identificados como CDs. Este evaluador identifica los contextos con estructuras más prototípicas de términos y definiciones.

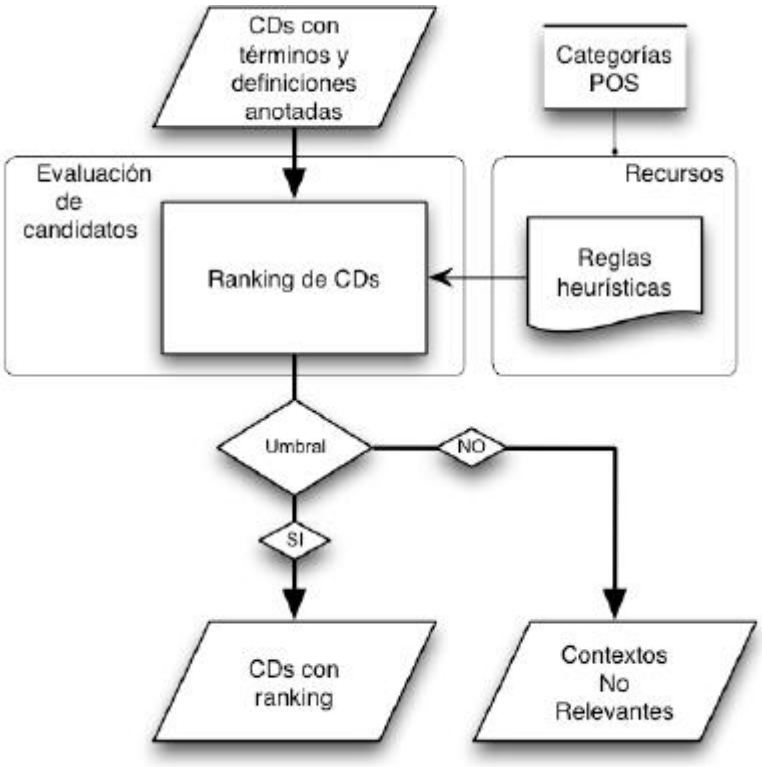

Figura 3.6 Evaluación de CDs (Alarcón, 2009:188)

Donde la entrada consiste en candidatos que fueron clasificados por el sistema como CDs. Para realizar la evaluación, el algoritmo usa algunas categorías POS para conformar un conjunto de reglas heurísticas que analizan la estructura sintáctica de los elementos automáticamente clasificados como términos o definiciones. Además de lo anterior, el algoritmo asigna valores numéricos para cada elemento constitutivo y estos valores se combinan en un elemento general que determina si cada CD pasa o no de acuerdo a un umbral preestablecido.

La salida final se condiciona: si el CD pasa el umbral, entonces se considera como un CD válido; si no, se considera como un contexto no relevante.

Las reglas heurísticas del procedimiento de evaluación se aprecian en la siguiente tabla:

Valor	Regla
Reglas de Término	
1	<t>$comillas.*$comillas</t>
1	<t>.*$parentesis $termino $parentesis</t>
2	<t>$coma.*$coma .*</t>
2	<t>$parentesis.*$parentesis .*</t>
3	<t>$demos\|$demos.*</t>
3	<t>.*$pron.*</t>
Reglas de definición	
1	<d>.* $que $verboConjugado</d>
1	<d2>$verboConjugado.*</d2>
2	<d>$palabra {,5}</d>
2	<d>.* (;|$noObstante| $sinEmbargo) .*</d>
3	<d>$demos</d>
3	<(d|d1)>NULL</(d|d1)>

Tabla 3.7 Reglas de evaluación de término y definición (Alarcón 2009: 189-191)

Se pueden observar diferentes reglas que asignan diversos valores a la estructura de términos y definiciones en la tabla 3.8. El valor 1 significa el mejor resultado, mientras que el 3 significa el peor; los candidatos que no siguen ninguna de las reglas son asignados con el valor 2. Por ejemplo, en el caso de estructuras de términos, el valor 1 es asignado a los casos que están marcados entre comillas, mientras que el valor 3 se asigna a aquellos candidatos cuya estructura de término consiste en un pronombre que pudiera indicar una posible referencia anafórica. En el caso de reglas de definición, el valor 1 es asignado a estructuras donde una clausula relativa se introduce después del pronombre "que", que puede ser una estructura prototípica en definiciones analíticas, mientras que un valor de 3 se asigna a los casos que solamente consisten en un pronombre demostrativo.

El algoritmo utiliza estas reglas para evaluar a cada candidato. Este proceso se realiza en tres pasos:

1. Evaluación de secuencias de términos y definiciones

    Para cada secuencia de término y definición en los candidatos

        Para cada regla de término y definición

            Si la secuencia de término y definición combinan con la regla

Etiquetar la secuencia con su valor correspondiente

2. Evaluar cada candidato con un valor global

Después de que todos los candidatos tienen su término y definición evaluados, el sistema asigna un valor global siguiendo las siguientes reglas:

Valor de evaluación de término y definición	Valor global de evaluación
<t = 1> y <d = 1>	<rG = 1>
<t = 2> o <d = 2>	<rG = 2>
<t = 2> y <d = 2>	<rG = 3>
<t = 3> o <d = 3>	<rG = 4>

Tabla 3.8 Reglas globales de evaluación de CDs

3. Excluir candidatos que no sobrepasan el umbral definido.

En este caso, he establecido el umbral en el valor 4.

Algunos ejemplos de los elementos de términos y definiciones evaluados son los siguientes:

Valor	**Ejemplo**
	Reglas de Términos
	<t v="1">«intrones»</t>
	<t v="3" >Este cloroplasto</t>
	Reglas de definición
	<t>la mutación rutabaga</t> <dvp>es </dvp> <d v="1">una mutación errónea que destruye a la adenilciclasa, interrumpiendo la síntesis del AMPc</d>.
	<d v="3">Esto</d> <dvp>se conoce <nx>como</nx></dvp> <t>mutación</t>.

Tabla 3.9 Ejemplo de términos y de definiciones evaluados (Alarcón 2009: 190,191)

De la tabla 3.10. Se puede observar que el ejemplo del término evaluado con 1 es valorado como un buen candidato por los marcadores tipográficos que enfatizan su presencia como un elemento importante en el texto; mientras que el ejemplo del término con valor de 3 es evaluado como el peor candidato porque el pronombre "este" se usa para hacer una referencia a un elemento que ha sido presentado anteriormente. Por otro lado, el primer ejemplo de las definiciones evaluadas se valora con un puntaje alto ya que tiene la

presencia del pronombre “que”, que está introduciendo la *differentia* en una definición analítica; finalmente, el segundo ejemplo se considera con valor 3 porque su estructura sintáctica que fue clasificada como una definición corresponde solamente a un pronombre relativo que está introduciendo una referencia anafórica.

Como salida se produce una lista de contextos definitorios etiquetados con sus partes distintivas en orden de calificación de su evaluación; esto consiste solamente en texto plano con etiquetas parecidas a XML. Un ejemplo de contexto etiquetado sería el siguiente:

<cd rG="1"><t>el chahuistle</t> <pvd><vd lema="ser">es</vd></pvd> <d>una enfermedad del maíz , conocida a la perfección por las comunidades prehispánicas que se dedicaban a la siembra de esta planta.</d></cd>

3.1.3.2. Problemas detectados

Se detectaron varios problemas en varios puntos del sistema: en las gramáticas, en el uso de estructuras de control y funcionalidades de Perl no utilizadas, redundancias en el código, en la entrada y la salida de datos, etc.

En las diferentes gramáticas se detectaron los siguientes problemas:

1. Las diferentes gramáticas se utilizan en diversos módulos sin embargo el sistema carga del disco duro la gramática cuando la requiere lo cual es ineficiente y debería quedar cargada desde el inicio del programa para su disposición de los diferentes módulos.
2. La gramática de categorías gramaticales contiene, además de la definición de las construcciones con etiquetas POS, una variable que define el inicio de línea; si bien esta variable puede ser utilizada para pasar información desde la línea de entrada hasta el CD de salida, es restrictiva y es utilizada a lo largo del programa para detectar el inicio de una línea de trabajo, considerando que Perl ya tiene dentro de sus expresiones regulares un operador que tiene esta función.

(01_gramaticaPOS.pm 19)

```
our $inicioLinea = "<doc_codi [^ ]*?>:";
```

3. La gramática de categorías gramaticales, que es un módulo de Perl, ejecuta un código que marca los tiempos verbales en palabras etiquetadas como verbos. Dicha ejecución debe hacerse en un módulo del sistema y no en la gramática.
4. La gramática de patrones verbales contiene una variable que controla si las raíces (expresiones regulares que permiten la definición de formas verbales) son consideradas únicamente como texto o como expresión regular; esta variable fue agregada al desarrollo por Alarcón (2009) para la corrección de posibles errores. La variable siempre debe estar activada para que busque los verbos con una expresión regular.

El siguiente es un fragmento de la gramática donde se define si la raíz es o no una expresión regular:

(Archivo: 02_gramPVDs.txt 22-24)

 # Aquí se definen si la raíz va o no seguida de cualquier carácter, excepto espacio en blanco ([^]*?) hasta el siguiente "/"

 # Se expresa en valores SI - NO

<raizRegex>si</raizRegex>

5. Las expresiones regulares que sustituyen los elementos de la gramática de filtros, en el módulo (Archivo: 04_filtro.pm) se cargan en memoria cada vez que se lee una regla del filtro, y estas expresiones son iguales en todas las reglas del filtro, por lo que se tienen que inicializar sólo una vez en el proceso.

En el código se detectaron otros problemas:

1. En el módulo de evaluación si bien la evaluación es automática, las reglas que se utilizan no pueden ser modificadas ni existe una gramática que las contenga, por lo que no se pueden agregar nuevas reglas, además existen varios tipos de reglas en los módulos cuya modificación podría ser de interés.
2. Se graba y lee un archivo para la comunicación interna entre cada uno de los archivos, lo que propicia que el sistema sea más lento y agregue tiempo de procesamiento en la entrada y salida a disco duro.
3. El código está escrito utilizando las estructuras de control ineficientemente, haciendo difícil la lectura del código y agregando líneas innecesarias o redundantes.

Por ejemplo: (Archivo: 03_td.pm 106-115).

```
1 #Si el candidato contiene NEXO
2 if (/<nx_[^ ]*?>/i)
3 {
4 }
5 # Si el candidato NO contiene NEXO
6 # ANOTA
7 else
8 {
9       if (/lema=\"$lemaDef\"/i)
10      {
11              s/($inicioLinea)/<tipoD=\"$tipoDef\"\/>$1/gi;
12      }
}
```

Aquí se ve como las líneas 2, 3 y 4 plantean una estructura condicional; sin embargo, es estéril su producción; su caso *default* (*else*) línea 7 a 13, es el que genera la sustitución. Adicionalmente el condicional de 9-12 sólo ejecuta una sentencia por lo que las llaves sobran. Además cabe mencionar que los comentarios en 5 y 6 interrumpen la lectura de la estructura del condicional. Esto debe ser arreglado para verse así:

```
 #Si el candidato contiene NEXO
2 if (! /<nx_[^ ]*?>/i) # Si el candidato NO contiene NEXO, Anota
3 {
4       s/($inicioLinea)/<tipoD=\"$tipoDef\"\/>$1/gi if (/lema=\"$lemaDef\"/i);
5 }
```

Se comprime la misma sustitución en 5 líneas en lugar de las 13 anteriores. Los comentarios no rompen la lectura de la sentencia y se comprime el segundo condicional, ya que sólo ejecuta una sola sentencia. Esto no ahorra ciclos de procesamiento, pero sí líneas de código difícil de leer.

4. Debido a que el árbol de decisión en el "Archivo: 05_arbol.pm" utiliza extensivamente estructuras condicionales, se genera un código confuso debido al mal aprovechamiento de las propias estructuras y de las expresiones regulares usadas en los condicionales. Un ejemplo de esto es (Archivo: 05_arbol.pm:725-739):

```
1 if (/<izq>NULL<\/izq>/i)
2 {
3 }
4 elsif (/<izq>$demos.*?<\/izq>/i)
5 {
6 }
7 elsif (/<izq><pvd/i)
8 {
9 }
10 else
11 {
12   s/(<izq>|<.izq>)//gi;
13   s/<der>/<izq>/gi;
14   s/(<izq>.*?) (<pvd.*?>.*?<.pvd.*?>) /$1<\/izq> $2 <der>/gi;
15 }
```

Debiendo quedar así:

```
1       if (! /<izq>((NULL|$demos.*)<\/izq>|<pvd)/i)
2       {
3                       s/(<izq>|<.izq>)//gi;
4                       s/<der>/<izq>/gi;
5                       s/(<izq>.*?) (<pvd.*?>.*?<.pvd.*?>) /$1<\/izq> $2 <der>/gi;
6       }
```

Lo que ahorra más de la mitad de las líneas de código haciendo el código más legible y de fácil lectura.

5 El módulo de reetiquetado (Archivo: 06_retagging.pm) idealmente no debería existir ya que el etiquetado pertinente debería hacerse en cada uno de los módulos y no necesitaría arreglarse el etiquetado.

6 Existen ciclos de procesamiento continuos uno después de otro cuya condición de iteración es idéntica y es posible combinarlos en un solo ciclo.

3.1.3.3. Soluciones propuestas

Las líneas de trabajo que se plantean para presentar un módulo más flexible y siguiendo los problemas detectados se enumeran a continuación.

Para las gramáticas se propone:

1. Crear un módulo nuevo que inicialice todas las variables globales y cargue en la memoria, de una sola vez, todas las gramáticas a utilizar para evitar repetir dicho proceso; además de permitir que el "Archivo: 01_gramaticaPOS.pm" sea exclusivamente un módulo de definición de variables derivadas de las etiquetas POS.
2. Eliminar la etiqueta de inicio de línea tanto en la gramática (Archivo: 01_gramaticaPOS.pm) como en todos los módulos subsecuentes en donde se emplea y, después, reemplazarla por el operador de inicio de línea de Perl en expresiones regulares "^".
3. Eliminar de la gramática de patrones verbales (Archivo: 02_gramPVDs.txt) la variable de prueba de expresiones regulares en las raíces verbales, fijando su valor en sí en los módulos correspondientes donde es usada.
4. Crear una gramática de reglas de evaluación.

Para el código en sí en los diferentes archivos se propone:

1. Analizar el código de cada uno de los módulos para quitar el uso de producciones inútiles en las estructuras de control, además de buscar expresiones regulares redundantes o que puedan ser simplificadas.

2. Eliminar toda lectura y escritura hacia el disco duro, en los módulos principales del sistema, es decir, pasar la información por memoria y no por archivo.
3. Analizar el funcionamiento de reetiquetado y tratar de incluirlo en las expresiones regulares de los módulos previos y posteriores.

3.1.4. Salida del Sistema

El sistema no posee una salida estructurada, originalmente entrega los CDs ordenados por su evaluación y en un archivo de texto uno por línea y con las partes que los conforman etiquetados e informando al usuario cuántos se obtuvieron.

3.1.4.1. Problemas detectados

La salida no es flexible, el usuario no puede recuperar la información que requiere ni permite hacer operaciones posteriores con los CDs. Además de que no permite la limpieza y filtrado de los CDs y sus etiquetas. Adicionalmente los usuarios pueden requerir como salida únicamente el texto segmentado de la entrada o solamente etiquetado con POS.

3.1.4.2. Soluciones propuestas

Implementar un módulo de salida que permita comunicar al usuario con la información recuperada por Ecode, que permita presentar los CDs al gusto del usuario y que sea capaz de comunicarse con los demás módulos así como con las interfaces de usuario.

El módulo tendrá la capacidad de procesar los CDs, limpiarlos y agruparlos por término semejante además de permitir conservar o eliminar etiquetas y presentar los CDs con un procesamiento posterior, todo en función de la facilidad de uso y de las salidas requeridas.

3.2. Empleo del Ecode.

Desde su desarrollo se ha propuesto al sistema Ecode como motor de más aplicaciones que pueden aprovechar a los contextos definitorios, entre las que destaca el Describe®. Por su parte, el Describe® es un sistema que permite buscar en internet documentos que contengan un término dado y PVDs; obtiene, por medio del Ecode, las definiciones de dicho término, además de presentar las definiciones agrupadas por posible homonimia. Este desarrollo se utiliza también como herramienta académica y de apoyo dentro del Grupo de Ingeniería Lingüística (GIL).

3.2.1. Página del Ecode.

Cuando Alarcón (2009) desarrolló el Ecode, a la par se creó una página descriptiva de lo que es Ecode en donde se tenía pensado crear una aplicación web del mismo; dicha posibilidad no llegó a materializarse. Se subieron análisis de un corpus en donde se tomaron como punto de partida los términos simples del Vocabulario Básico del Genoma Humano del Corpus Técnico del IULA (http://www.iula.upf.edu/rec/vbgenoma/esp/frames.html). Se extrajeron los CDs de textos de términos del genoma humano y se agruparon por tipo de definición y además por PVD, indicando cuántos CDs se extrajeron y cuántos candidatos fueron filtrados.

Tipo de Definición	Patrón Verbal	CDs	No Relevantes
Genus y Diferencia	ser + det + N	7	52
Genus y Diferencia	definir como	5	5
Genus y Diferencia	entender como	1	4
Genus y Diferencia	concebir como	0	0
Genus y Diferencia	identificar como	0	32
Extensional	constar de	3	11
Extensional	formar por	7	27
Extensional	contener	1	4
Extensional	Tener	2	12
Funcional	Usar en	0	17
Funcional	Usar como	0	8
Funcional	Usar para	0	18
Funcional	Utilizar en	1	41

Tabla 3.10 Resultados de GEN por Tipo de definición y PVD en la página de Ecode (http://www.iula.upf.edu/rec/vbgenoma/esp/frames.html)

Además permite consultar los contextos extraídos de cada término por PVD. Cabe mencionar que dichos resultados fueron procesados y sólo se subió una interfaz para consultarlos, de ninguna manera está ligado el sistema Ecode con dicha página y fue pensada únicamente para informar y difundirlo[11].

3.2.2. Describe

Describe es un proyecto desarrollado en el Grupo de Ingeniería Lingüística, patrocinado a través de un proyecto CONACyT, que lleva el nombre de "Extracción de relaciones léxicas para dominios restringidos a partir de contextos definitorios en Español".

El sistema Describe consiste en un conjunto de módulos que, a través de una interfaz web, reciben un término y lo buscan por medio de un motor de búsqueda incluyendo el término y los patrones verbales definitorios. De los URLs resultantes de la búsqueda, se descarga el texto de los documentos.

[11] Su URL es: http://brangaene.upf.es/ecode

Ecode es el motor principal de Describe, utiliza los documentos extrayéndoles el texto, se obtienen los CDs y se filtran por término para poder entregar un conjunto de CDs agrupados por definición.

Después, los contextos extraídos se agrupan, en otro módulo, por similitud de las definiciones y se presentan al usuario en una interfaz web.

El Ecode cumple la función de núcleo del proyecto Describe; si bien en este momento los módulos que constituyen a Describe no se encuentran plenamente integrados, el Ecode es parte integral de éste.

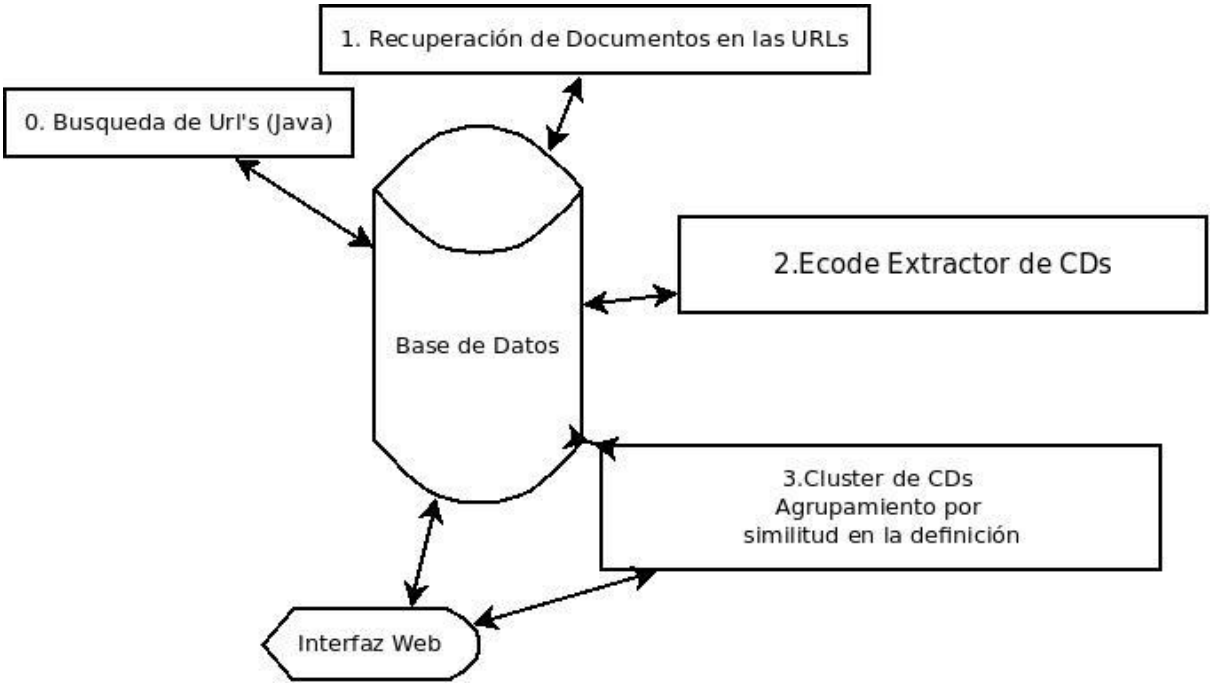

Figura 3.7 Esquema del sistema Describe

El flujo de un término hasta encontrar sus CDs y presentarlos al usuario final se representa en la siguiente figura.

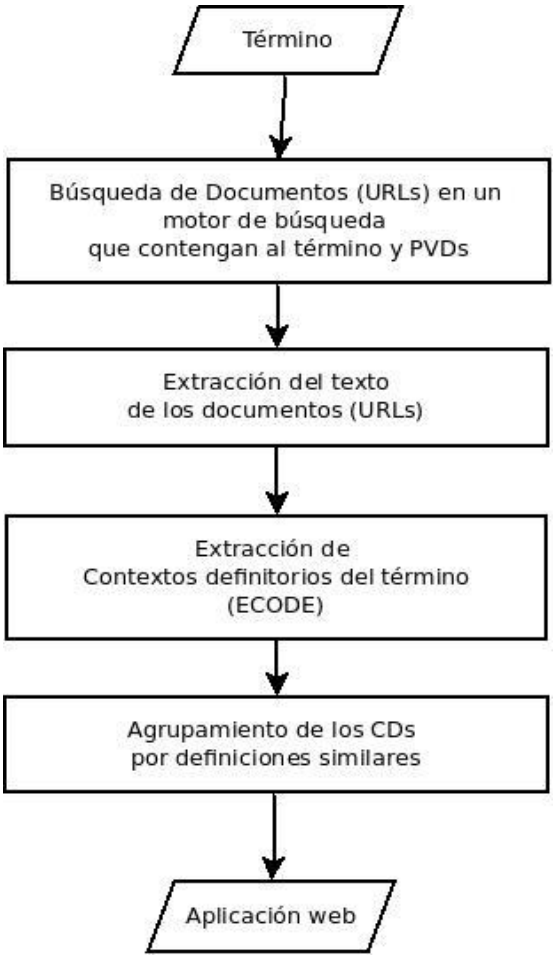

Figura 3.8 Flujo del Sistema Describe

Describe tiene su propio sitio web[12] que cuenta con alrededor de 500 términos agrupados y listos para ser presentados.

3.3. Recapitulación de problemas y líneas de trabajo

En general se puede decir que el sistema Ecode presenta 3 problemas principales:

1. No es flexible en la entrada y requiere de un preprocesamiento que no es sencillo y no tiene las herramientas adecuadas.

[12] http://www.describe.com.mx:8080/describeRe/15/

Se propone el desarrollo de un módulo de entrada que preprocese el texto, lo separe por oraciones, lo etiquete con sus categorías gramaticales, haga un filtrado previo y presente una entrada lista para Ecode.

2. Los módulos originales presentan varios problemas: primero, las estructuras de control no fueron utilizadas adecuadamente y existen archivos intermedios entre los módulos lo cual lo hace más lento. Además tiene una gran cantidad de variables globales que, o se usan una sola vez, o ya fueron declaradas con otro nombre en otro módulo. Por otra parte, las gramáticas presentan variables innecesarias o que no requieren formar parte de ellas, además de que se necesita de la posibilidad de poder agregar más reglas a ellas.

La solución que se propone es reescribir el código en el que las estructuras de control no se utilizaron correctamente, eliminar los accesos a disco duro intermedios entre los módulos, y eliminar variables excesivas o declaradas como globales.

Modificar las gramáticas, limpiándolas y permitiendo únicamente la existencia de variables estrictamente requeridas, además de admitir que el usuario agregue reglas en todas y cada una de ellas.

3. La salida del sistema únicamente es una lista de contextos y no contiene ningún procesamiento posterior (agrupamiento, lematización, conexión a bases de datos, limpieza de etiquetas, salida en XML o una aplicación web).

Para proveer funcionalidades que permitan dar formato a los CDs de salida es necesario crear un módulo de salida que permita hacer una serie de operaciones para la interconexión con otros sistemas y la presentación clara al usuario, así como una aplicación web.

Estos son los principales problemas y las propuestas básicas para optimizar y desarrollar el sistema Ecode. En el siguiente capítulo se tratarán las líneas de trabajo planteadas y sus soluciones implementadas así como su posible desarrollo futuro.

4. Desarrollo y trabajo sobre el sistema Ecode

Se propusieron algunas soluciones para optimizar el sistema Ecode que, después de analizar su viabilidad y conveniencia, fueron implementadas. En general se desarrollaron dos módulos adicionales dedicados a la entrada y salida de los datos, además de hacer una limpieza general del código y reescritura de las expresiones regulares.

El sistema original quedó asimilado por completo en el nuevo sistema y no cambió el flujo de los datos dentro de éste, a pesar de haber sido reescrito parcialmente y optimizado. El modelo final del Ecode se muestra en la siguiente figura:

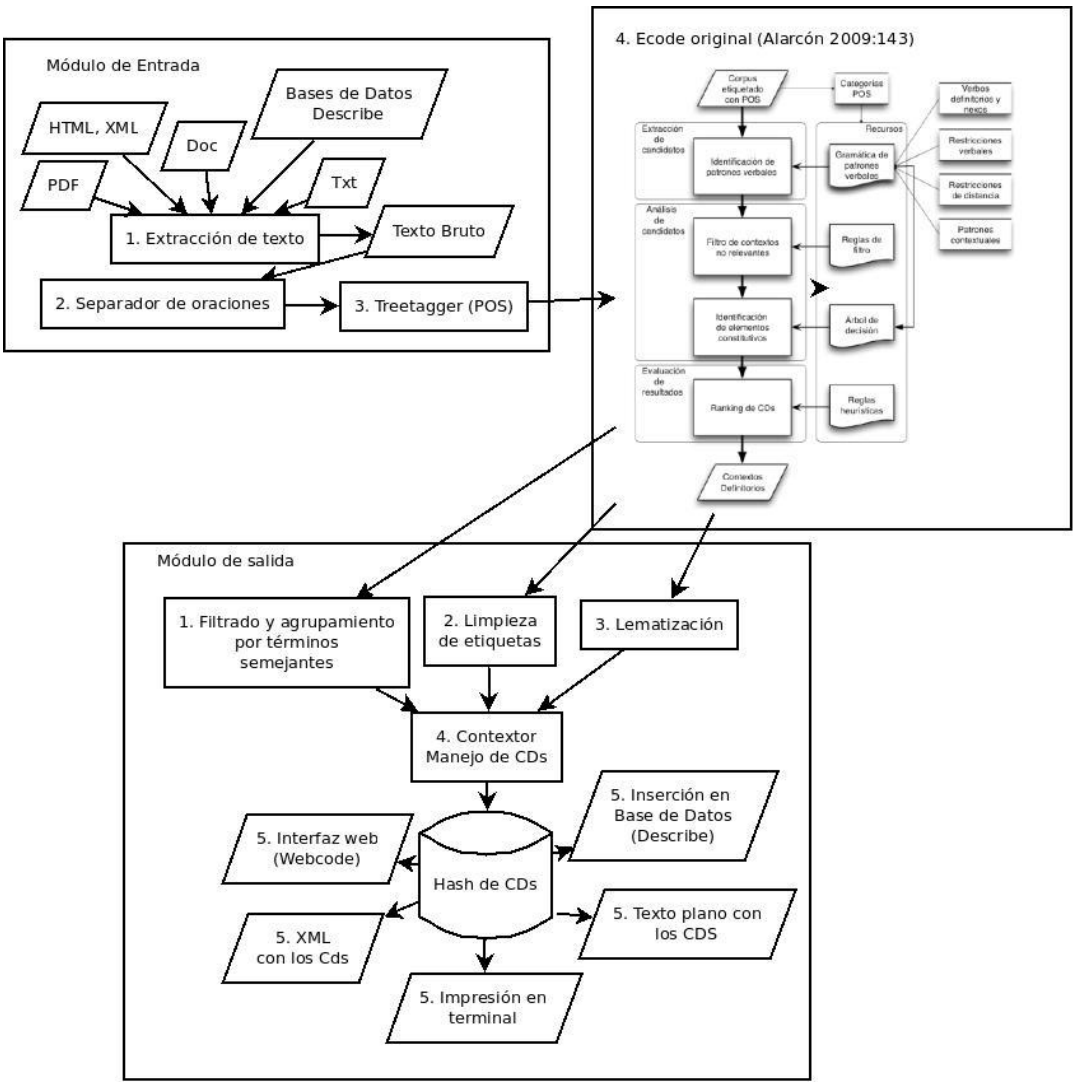

Figura 4.1 Panorama final del Ecode

Como se puede ver en la figura anterior, la adaptación del sistema se basó, fundamentalmente, en flexibilizar las entradas de datos además del acoplamiento con el sistema Describe. Por una parte, la optimización se realizó a nivel de código y de expresiones regulares del Ecode original; mientras que la expansión se llevó a cabo proponiendo una estructura de datos que entregue fácilmente los CDs previamente extraídos y permita otras funcionalidades como lematizar los contextos y limpiar sus etiquetas.

Además de los archivos con los que cuenta originalmente el sistema, se agregaron archivos en donde se contienen los módulos de entrada y salida. Veamos con mucho más detalle, en la siguiente tabla, los nuevos archivos que se integraron al sistema original:

Archivo	**Descripción**
00_Inicia.pm	Inicializa variables globales, carga y preprocesa gramáticas.
ES_entrada.pm	Extracción de texto, etiquetador POS, separador de oraciones.
ES_salida.pm	Eliminación de etiquetas, lematización, creación de Hash de CDs y filtrado por términos.
ES_DB_describe.pm	Funcionalidad para trabajar con la base de datos de Describe.
ES_File_Type.pm	Módulo de Perl para identificar el tipo de archivo de entrada.
Porter_Stem_Es.pm	Lematizador Porter para el español.
util.pm	Funciones utilizadas a lo largo del sistema como apoyo.
libecode.pm	Permite la ejecución secuencial del módulo de entrada, Ecode y salida.

Tabla 4.1 Archivos agregados a Ecode

4.1 Optimización del Ecode original

La optimización del módulo central, es decir, el Ecode original, se realizó en tres etapas. Después de un análisis del flujo del programa se encontró que lo que debía optimizarse eran las operaciones de entrada y salida entre los módulos de procesamiento del Ecode, el uso indebido y extenso de las estructuras de control y de variables globales, y la utilización de expresiones regulares de forma inapropiada o sin aprovechar el potencial de éstas.

4.1.1 Operaciones de Entrada/Salida

El Ecode original presentaba lectura y escritura a disco duro entre cada archivo, lo cual, si bien sirve para obtener información intermedia del procesamiento, lo ideal sería almacenar dicha información y realizar una sola operación de escritura al final del procesamiento disco.

Por ejemplo (Archivo: 01_vd.pm 156-164):

```
# Imprime un documento con todas las líneas que tienen un verbo definitorio en la carpeta de OUTPUT
open FILE, (">outputAux/s01_preCandidatos.txt") or die "No se puede escribir el documento con preCandidatos!";
print FILE our @preCandidatos;
close FILE;

# Imprime un documento con todas las líneas que NO tienen un Verbo Definitorio en la carpeta de OUTPUT
open FILE, (">outputAux/n01_preNRsSinVDs.txt") or die "No se puede escribir el documento con n01_preNRsSinVDs!";
print FILE our @NRs1;
close FILE;
```

Este fenómeno se repite a lo largo de cada archivo y para pasar la información se utilizan arreglos globales que pueden suplir directamente en la memoria la tarea de los archivos sin tener que recurrir a la escritura en el disco duro. La siguiente tabla muestra cuántas lecturas y escrituras de archivos se producen en el sistema:

Archivo (Módulo)	Operaciones de Entrada –(lectura de archivos)	Operaciones de Salida (Escritura de archivos)
01_vds.pm	1	2
02_pvds.pm	2	2
03_td.pm	2	1
04_filtro.pm	3	2
05_arbol.pm	2	2
06_retagging.pm	1	1
07_rankingTyD.pm	2	1
08_rankingGlobal.pm	2	1
09_final.pm	7	6
01_gramaticaPOS.pm	0	0
ecode.pl	1	3

Tabla 4.2 Ocurrencias de operaciones de entrada/salida en los módulos de Ecode

De la tabla anterior salta a la vista que la escritura y lectura se llevan a cabo en cada archivo para pasar al arreglo de información a través del sistema. Las lecturas restantes pertenecen a lectura de las gramáticas, realizándose repetidas veces en los módulos, lo que resulta redundante y probablemente implique que sean cargadas solamente una vez. Además de lo anterior, las escrituras restantes representan archivos de procesamiento intermedio para motivos de corrección de errores, esta funcionalidad se mantuvo durante el desarrollo de la optimización para la realización de pruebas; sin embargo, el usuario final será capaz de decidir si requiere o no esta información.

4.1.2 Simplificación de código

El código del Ecode original presentaba un uso ineficiente de las estructuras de control y realizaba varios procesamientos previamente realizados en módulos anteriores, por lo que se optó por simplificar el código con las siguientes consideraciones:

4.1.2.1 Inicialización de variables y gramáticas

En primer lugar, se creó el módulo (00_inicia.pm) que inicializa las variables globales que van a ser usadas (arreglos de gramáticas, arreglos de paso de información entre módulos, etc.). En este paso, se preprocesan y cargan las gramáticas en arreglos globales para ser llamadas desde la memoria en los diferentes módulos. El preprocesamiento consiste en separar la información de las gramáticas y colocarla en arreglos.

4.1.2.2 Simplificación de variables y su alcance

En todos los módulos se intentaba saber cómo fueron declaradas las variables y su uso interno; con esto, se descubrió que en todos los casos las variables fueron declaradas como globales en Perl, lo que podría propiciar un conflicto entre nombres de variables, además de que resulta innecesario tener en la memoria toda la información de procesos anteriores, pues ya no es relevante su existencia. Adicionalmente se encontró que el operador *our* era usado para cada aparición de las variables; si bien esto no genera algún error por parte del intérprete de Perl, el proceso equivale a declarar muchas veces las variables, razón por la cual fue corregido. Veamos el siguiente ejemplo que aclara más el asunto:

(Archivo Ecode original: 02_pvds.pm: 89-105):

```
foreach our $patron (our @arrayPVD1)
{
if ($patron =~ /<lm>(.*?)<.lm>/)
{
                    our $lema = $1;
}
our $lema;
our $vd = "<vd lema=\"$lema\">[^]*?<.vd>";
foreach our $preCandidato (@preCandidatos)
{
# PVD - Rule 05
while ($preCandidato =~ / <id[^]*?>$vd /gi)
{
```

```
$preCandidato =~ s/ <id([^ ]*?)>($vd) / pvd_r05_$1><vd_$1>$2<\/vd_$1><\/pvd_r05_$1> /i;
}
}
}
```

En el ejemplo se aprecia el uso extensivo del operador *our* que declara variables, pero se utiliza en cada aparición de la variable lo cual resulta ser confuso e innecesario. El problema se reparó de esta manera (Archivo Ecode final: 02_pvds.pm: 18-26 76-86):

```
our (@gramaPVDs,@patrones,@preCandidatos,@candidatos,@NRs);
our ($sele, $se, $tag,$aux,$verboCon);
my (@arrayPVD1, @arrayPVD2,@arrayPVD3);
my ($patron,$lema,$vd,$nexoAux,$nexo,$preCandidato,$distancia,$distanciaAux);
…
for $patron (@arrayPVD1)
{
$lema = $1 if ($patron =~ /<lm>(.*?)<.lm>/) ;

        $vd = "<vd lema=\"$lema\">[^ ]*?<.vd>";
        for $preCandidato (@preCandidatos)
        {
                # PVD - Rule 05
$preCandidato =~ s/ <id([^ ]*?)>($vd) / <pvd_r05_$1><vd_$1>$2<\/vd_$1><\/pvd_r05_$1> /I
while ($preCandidato =~ / <id[^ ]*?>$vd /gi) ;
}
}
```

En el ejemplo anterior se puede apreciar a las variables que quedan declaradas al inicio del módulo, separadas por globales y locales, y que posteriormente son usadas sin el operador *our* en el módulo, además se observa de qué manera se comprimieron las estructuras de control.

Otro punto interesante, es que no se utilizó el paso de información entre las diferentes funciones por medio de sus argumentos, simplemente se instanciaba la variable requerida con el operador *our* y se utilizaba, por lo que se implementó el paso de argumentos entre las diferentes funciones.

4.1.2.3 Reescritura de expresiones regulares

Para el mejoramiento del sistema Ecode original, se hicieron cambios en algunas expresiones regulares, por ejemplo: se eliminó la variable *$iniciolinea* que marcaba el inicio de una línea, pues las expresiones regulares proveen esta funcionalidad, además de utilizar los operadores de las expresiones regulares para simplificar condicionales y bucles en el sistema. Un ejemplo de la reescritura de expresiones regulares es ilustrado en el siguiente ejemplo (Archivo Ecode Original: 05_arbol.pm: 465-482):

```
if (/<izq>.*? \(\/[^]*? ($termino1.*?)<.izq>/i)
{
        if ($1 =~ /\)\/[^]*?/i)
        {
        }
        elsif ($1 =~ /$verboCon/i)
        {
        }
        else
        {
                s/<izq>(.*? \(\/[^]*?) ($termino1.*?)<.izq>/$1 <t>$2<\/t>/gi;
                s/<der>(.*?) (\)\/[^]*? .*?)<.der>/<d>$1<\/d> $2/gi;
                s/<der>(.*?) (\)\/[^]*?)<.der>/<d>$1<\/d> $2/gi;
                s/<der>(\)\/[^]*? .*?)<.der>/<d>$1<\/d>/gi;
                s/<der>(\)\/[^]*?)<.der>/<d>$1<\/d>/gi;
                s/(<cand.>)/$1<cd_RI16\/>/gi;
        }
}
```

En esta secuencia se observa claramente el uso erróneo de la estructura de control condicional, por lo que es posible simplificarla uniendo los 2 primeros condicionales en uno solo reescribiendo la estructura de control y la expresión regular, de esta manera (Archivo Ecode Final: 05_arbol.pm: 430-441):

```
if (/<izq>.*? \(\/[^]*? ($termino1.*?)<.izq>/i)
{
        if ($1 !~ /\)\/[^]*?|$verboCon/i)
        {
                s/<izq>(.*? \(\/[^]*?) ($termino1.*?)<.izq>/$1 <t>$2<\/t>/gi;
                s/<der>(.*?) (\)\/[^]*? .*?)<.der>/<d>$1<\/d> $2/gi;
                s/<der>(.*?) (\)\/[^]*?)<.der>/<d>$1<\/d> $2/gi;
                s/<der>(\)\/[^]*? .*?)<.der>/<d>$1<\/d>/gi;
                s/<der>(\)\/[^]*?)<.der>/<d>$1<\/d>/gi;
                s/(<cand.>)/$1<cd_RI16\/>/gi;
        }
}
```

El procedimiento seguido para el mejoramiento del sistema en este punto en particular fue el siguiente: se reescribió la expresión regular del condicional englobando las 2 condicionales que no producían ningún procesamiento y negándolas, con esto se simplificaría enormemente el código y se haría más rápido, ya que genera menos comparaciones.

4.1.2.4 Compresión de estructuras de control

Por otra parte, se decidió comprimir las estructuras de control, en especial los condicionales, ya que en su uso extensivo no se tomaron en cuenta los operadores que permite Perl para comparar entre las expresiones regulares. Durante el proceso se consideraron diferentes formas de comprimir las estructuras de control: primero, se comprimieron todas aquellas estructuras de control (*if, for, while*) que sólo produjeran una línea; esto puede considerarse meramente por estilo, sin embargo, se realizó por simplificación.

Por ejemplo, el Archivo Ecode Original: 03_td.pm: 62-65 a continuación ilustrado:

```
if ($patron=~ /<nx>(.*?)<.nx>/i)
{
        our $nexoDef = $1;
        }
```

Se comprimió en (Archivo Ecode Final: 03_td.pm: 39):

```
$nexoDef = $1 if ($patron=~ /<nx>(.*?)<.nx>/i);
```

Además, en el caso de los condicionales *if* se comprimieron los que presentaban su producción principal y un *elsif* que producía sentencias, por ejemplo, el Archivo Ecode Original: 03_td.pm: 85-91:

```
if (/<nx_$lemaID>/i)
{
}
else
{
        s/($inicioLinea)/<tipoD=\"$tipoDef\"\/>$1/gi;
}
```

Se comprimió en el Archivo Ecode Final: 03_td.pm: 59:

```
        s/^/<tipoD=\"$tipoDef\"\/>/gi  if (! /<nx_$lemaID>/i) ;
```

Asimismo, existen estructuras condicionales que presentan varias condiciones anidadas que no producen nada y su producción es en el *else*. Retomando un ejemplo anterior (Archivo Ecode Original: 05_arbol.pm: 465-482), podemos ver la situación de manera concreta:

```
if (/<izq>.*? \(\/[^ ]*? ($termino1.*?)<.izq>/i)
{
        if ($1 =~ /\)\/[^ ]*?/i)
        {
```

```
        }
        elsif ($1 =~ /$verboCon/i)
        {
        }
        else
        {
                s/<izq>(.*? \(\/[^]*?) ($termino1.*?)<.izq>/$1 <t>$2<\/t>/gi;
                s/<der>(.*?) (\)\/[^]*? .*?)<.der>/<d>$1<\/d> $2/gi;
                s/<der>(.*?) (\)\/[^]*?)<.der>/<d>$1<\/d> $2/gi;
                s/<der>(\)\/[^]*? .*?)<.der>/<d>$1<\/d>/gi;
                s/<der>(\)\/[^]*?)<.der>/<d>$1<\/d>/gi;
                s/(<cand.>)/$1<cd_RI16\/>/gi;
        }
}
```

Se sustituyó reescribiendo la expresión regular y comprimiendo el condicional (Archivo Ecode Final: 05_arbol.pm: 430-441), de la manera siguiente:

```
if (/<izq>.*? \(\/[^]*? ($termino1.*?)<.izq>/i)
{
        if ($1 !~ /\)\/[^]*?|$verboCon/i)
        {
                s/<izq>(.*? \(\/[^]*?) ($termino1.*?)<.izq>/$1 <t>$2<\/t>/gi;
                s/<der>(.*?) (\)\/[^]*? .*?)<.der>/<d>$1<\/d> $2/gi;
                s/<der>(.*?) (\)\/[^]*?)<.der>/<d>$1<\/d> $2/gi;
                s/<der>(\)\/[^]*? .*?)<.der>/<d>$1<\/d>/gi;
                s/<der>(\)\/[^]*?)<.der>/<d>$1<\/d>/gi;
                s/(<cand.>)/$1<cd_RI16\/>/gi;
        }
}
```

Al final, por estilo se sustituyeron las estructuras de control *foreach* por *for*, ya que son equivalentes y tienen la misma sintaxis[13].

4.1.2.5 Conjunción de ciclos similares

Durante el proceso experimental, resultó muy frecuente que se presentaran situaciones en las que un ciclo *for* era usado dos o más veces consecutivas para iterar un arreglo y realizar operaciones sobre él, situación que podría optimizarse simplificando y conjuntado varios ciclos *for* en uno solo. Con este proceso se ahorran muchos ciclos de procesamiento y se simplifica el código. Veamos un ejemplo (Archivo Ecode Original: 02_pvds.pm: 169-237):

```
# Sustituye <vd_#></vd_#> por <vd_# lema.*?></vd_#>
foreach (@preCandidatos)
{
        # Sustituye <vd_ID><vd lema=".*"> por <vd_ID lema=".*">
        s/<vd(_[^]*?)>(<vd)( lema)/$2$1$3/gi;
        s/<\/vd>(<\/vd_[^]*?>)/$1/gi;
        # Limpia dobles espacios entre <\/vd> y <nx>
        s/  (<nx)/ $1/gi;
}
# Sustituye etiqueta AUXILIAR: <prAux_#></prAux_#>
foreach (@preCandidatos)
{
        our ($sele, $aux);
s/<prAux_([^ ]*?)>($sele) ($aux)<.prAux_[^ ]*?>/<pr_$1>$2<\/pr_$1>
<aux_$1>$3<\/aux_$1>/gi;
        s/<prAux_([^]*?)>($sele)<.prAux_[^]*?>/<pr_$1>$2<\/pr_$1>/gi;
        s/<prAux_([^]*?)>($aux)<.prAux_[^]*?>/<aux_$1>$2<\/aux_$1>/gi;
        # Añade etiqueta SE: <\/vd> se <NX>
        s/(<.vd_)([^]*?)(>) ($sele) (<nx_[^]*?>)/ $1$2$3 <pr_$2>$4<\/pr_$2> $5/gi;
}
```

[13] (http://perldoc.perl.org/perlsyn.html#Foreach-Loops)

```
# Expande etiqueta de PVD
foreach (@preCandidatos)
{
	our ($sele, $se, $aux);
	# Expande <pvd_r..> a la IZQUIERDA (excepto para PVDs con lema SER)
	while (/ ((?:$sele $aux|$sele|$aux) <pvd_r.._[^]*?><vd_[^]*? [^]*?>)/gi)
	{
		if ($1 =~ /<vd_[^]*? lema=\"ser\">/)
		{
		}
		else
		{
			# SELE + AUX
s/ ($sele) ($aux) (<pvd_r.._)([^ ]*?)(>)(<vd_[^ ]*? [^ ]*?>)/ $3$4$5<pr_$4>$1<\/pr_$4>
<aux_$4>$2<\/aux_$4> $6/gi;
			# SELE
s/ ($sele) (<pvd_r.._)([^]*?)(>)(<vd_[^]*? [^]*?>)/ $2$3$4<pr_$3>$1<\/pr_$3> $5/gi;
			# AUX
s/ ($aux) (<pvd_r.._)([^]*?)(>)(<vd_[^]*? [^]*?>)/ $2$3$4<aux_$3>$1<\/aux_$3> $5/gi;
		}
	}
	# Expande <pvd_r..> a la DERECHA
	# SELE
	s/(<.vd_[^]*?>)(<.pvd_r.._)([^]*?)(>) ($sele) /$1 <pr_$3>$5<\/pr_$3>$2$3$4 /gi;
}

# ------------------------------------------------->>>>>>>>
# PUSH + WARNING
# ------------------------------------------------->>>>>>>>
foreach (@preCandidatos)
{
	our $KpreCandidatos++;
	# Push candidatos
	if (/(<pvd_[^]*?>)/)
```

```
        {
                push our @candidatos, ($_);
        }
        else
        {
                # Push No Relevantes 2 - Sin Patrones Verbales
                push our @NRs2, ("<NRs tipo=\"sinPVDs\"\/>".$_);
                # Comprueba que no queden verbos definitorios anotados sin procesar
                if (/<.vd_[^]*?>/)
                {
                        push our @warnings, ("Faltan vds por procesar - 02_pvds.pm linea 163\n");
                }
        }
}
```

Si se mira con detenimiento en el ejemplo anterior se aprecia como un ciclo *foreach* se repite cuatro veces consecutivas que itera sobre *@preCandidatos*, esto se puede comprimir de la siguiente manera, para obtener mayor eficiencia en el proceso y un mayor ahorro de espacio (Archivo Ecode Final 02_pvds.pm: 135-176):

```
# Sustituye <vd_#></vd_#> por <vd_# lema.*?></vd_#>
for (@preCandidatos)
{
        # Sustituye <vd_ID><vd lema=".*"> por <vd_ID lema=".*">
        s/<vd(_[^]*?)>(<vd)( lema)/$2$1$3/gi;
        s/<\/vd>(<\/vd_[^]*?>)/$1/gi;
        # Limpia dobles espacios entre <\/vd> y <nx>
        s/  (<nx)/ $1/gi;
        # Sustituye etiqueta AUXILIAR: <prAux_#></prAux_#>
s/<prAux_([^ ]*?)>($sele) ($aux)<.prAux_[^ ]*?>/<pr_$1>$2<\/pr_$1><aux_$1>$3<\/aux_$1>/gi;
        s/<prAux_([^]*?)>($sele)<.prAux_[^]*?>/<pr_$1>$2<\/pr_$1>/gi;
        s/<prAux_([^]*?)>($aux)<.prAux_[^]*?>/<aux_$1>$2<\/aux_$1>/gi;
```

```
        # Añade etiqueta SE: <\/vd> se <NX>
        s/(<.vd_)([^]*?)(>) ($sele) (<nx_[^]*?>)/ $1$2$3 <pr_$2>$4<\/pr_$2> $5/gi;

        # Expande etiqueta de PVD<pvd_r..> a la IZQUIERDA (excepto para PVDs con lema SER)
        while (/ ((?:$sele $aux|$sele|$aux) <pvd_r.._[^]*?><vd_[^]*? [^]*?>)/gi)
        {
                if ($1 !~ /<vd_[^]*? lema=\"ser\">/)
                {
                        # SELE + AUX
s/ ($sele) ($aux) (<pvd_r.._)([^ ]*?)(>)(<vd_[^ ]*? [^ ]*?>)/ $3$4$5<pr_$4>$1<\/pr_$4>
<aux_$4>$2<\/aux_$4> $6/gi;
                        # SELE
s/ ($sele) (<pvd_r.._)([^]*?)(>)(<vd_[^]*? [^]*?>)/ $2$3$4<pr_$3>$1<\/pr_$3> $5/gi;
                        # AUX
s/ ($aux) (<pvd_r.._)([^]*?)(>)(<vd_[^]*? [^]*?>)/ $2$3$4<aux_$3>$1<\/aux_$3> $5/gi;
                }
        }
        # Expande <pvd_r..> a la DERECHA  SELE
        s/(<.vd_[^]*?>)(<.pvd_r.._)([^]*?)(>) ($sele) /$1 <pr_$3>$5<\/pr_$3>$2$3$4 /gi;

        # Push candidatos
        if (/(<pvd_[^]*?>)/)
        {
                push @candidatos, ($_);
        }
        else
        {
                # Push No Relevantes 2 - Sin Patrones Verbales
                push @NRs, ("<NRs tipo=\"sinPVDs\"\/>".$_);
        }
}
```

Al final, solamente en este ejemplo se pudieron comprimir cuatros ciclos en uno solo.

4.1.3 Impacto de la optimización del Ecode original y trabajo futuro

La optimización realizada al sistema es el primer paso para convertir al Ecode, no sólo en un sistema eficiente, sino también en una fuente para la investigación de contextos definitorios que fue concebida para solucionar problemas computacionales no resueltos y como un apoyo para los lingüistas en sus estudios del léxico.

Como ya hemos explicado, se reescribió gran parte del código del sistema, se revisaron las expresiones regulares y se planteó un esquema de trabajo para simplificar el código y arreglar problemas detectados. Asimismo, se redujeron, en gran medida, el número de líneas del código y la cantidad de ciclos de procesamiento que se requieren para ejecutarlo, lo que se tradujo en un sensible mejoramiento del sistema.

El siguiente paso en la optimización de Ecode consistía en permitir a los usuarios modificar las gramáticas y agregar nuevas reglas, esto mediante una interfaz que convierte las gramáticas a XML; ya que existen reglas importantes que deberían estar en gramáticas y, además, deberían ser configurables (por ejemplo, las reglas de evaluación de los CDs (Ranking)).

Las secuencias de expresiones regulares deben ser analizadas una por una para tratar de simplificar el conjunto basado en las entradas y salidas que requiere cada módulo, es decir, que en vez de revisar cada expresión regular una por una, se analizarían como un todo; como si fueran una gramática y, al mismo tiempo, tratar de simplificar la cantidad de expresiones regulares totales.

4.2 El Módulo de entrada de datos

En un principio el Ecode recibía de entrada un texto previamente separado por oraciones y etiquetado con POS, por lo que se tuvo que generar un módulo completo para producir esa entrada a partir de los archivos de aplicación e internet, para leer o producir texto, más frecuentes entre los usuarios de computadoras de hoy en día (Doc, HTML, PDF, XML).

Fue necesario encontrar un nuevo etiquetador POS para adecuarlo al sistema, ya que el etiquetador disponible era muy lento. Entonces, se adaptó el TreeTagger (véase 2.1.3.2) al sistema y se desarrolló una interfaz que permitiera dicho acoplamiento.

El módulo de entrada se puede apreciar en la siguiente figura:

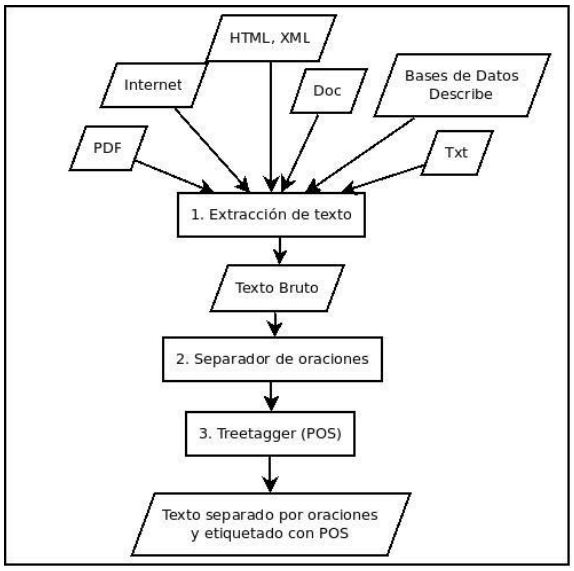

Figura 4.2 El módulo de entrada de Ecode

4.2.1 La fuente de los datos

El uso del sistema estaba restringido a llevar una serie de procesos previos al Ecode, anteriormente mencionados; sin embargo, uno de los objetivos de esta tesis es desarrollar un sistema que acoplara Ecode al sistema Describe como motor principal, además de presentar una herramienta independiente a usuarios que requieran extraer información de sus propios textos. Por lo tanto, fue necesario desarrollar una interfaz independiente para el usuario interesado únicamente en el Ecode y los usuarios que generaran una ejecución de Ecode desde el sistema Describe.

4.2.1.1 Texto plano y extracción de éste a partir de archivos de aplicación de usuario e internet

La conexión a la base de datos de Describe era un requerimiento importante del Ecode; sin embargo, también fue requerida una herramienta que permitiera extraer los CDs de textos y orígenes particulares a usuarios que los requirieran.

Los usuarios concentran su lectura y escritura de texto en formatos de aplicación (Doc, pdf, html, xml), por lo que lo más natural para un usuario final es mantener su corpus textual en alguno de éstos formatos. Algunos usuarios podrían utilizar texto plano (txt), lo cual también está contemplado, y adicionalmente internet; por lo tanto, se pensó en desarrollar

un módulo de entrada flexible que se ajustara a las necesidades de entrada de los usuarios.

Para la identificación del tipo de archivo se utilizó el módulo estándar File::type[14] de Perl, que recibe el archivo, lee su cabecera e identifica el tipo de archivo que es; este módulo se utiliza para poder discernir entre el método de extracción del texto de acuerdo con el tipo de archivo.

En la siguiente tabla se describen los tipos de archivo de aplicación que soporta Ecode y la herramienta que se usa para extraer el texto de ellos:

Tipo de archivo de aplicación	Herramienta de extracción de texto
Documento de Word (Doc) hasta la versión 2003	Antiword, herramienta de extracción de texto de archivos de Word
Documento portátil (PDF)	Cam::PDF, Módulo de manejo de PDFs
Lenguajes de etiquetado (XML, HTML, XHTML)	Elinks, explorador web de modo texto con capacidad de volcar el texto
Texto plano (txt)	Perl, Ecode
Todos los archivos de un directorio	Perl
Internet	Elinks, explorador web de modo texto con capacidad de volcar el texto

Tabla 4.3 Herramientas para la extracción de texto

Todas estas herramientas convergen en el flujo del programa entregando el texto bruto que extrajeron del tipo de archivo correspondiente.

4.2.2. El problema de las codificaciones

Uno de los problemas que llevó más tiempo resolver, amén de que no se pudo llegar a una solución total, fue el de las codificaciones de entrada al sistema. Si bien el texto en español suele escribirse sólo en dos o tres codificaciones de caracteres, en la práctica no es así, ya que frecuentemente se encuentran caracteres mal formados o no reconocidos, errores de codificación, mezcla entre codificaciones y errores en el reconocimiento de éstas.

Así que se realizó un análisis y se encontró que la gran mayoría de texto en español se

[14] (http://search.cpan.org/dist/File-Type/lib/File/Type.pm)

encuentra o en UTF-8 o en ISO-8859-1, lo cual permitió buscar herramientas que detectaran y cambiaran las codificaciones de los documentos de entrada.

Además, se presentó la problemática de que un texto podría tener varias codificaciones, en especial viniendo de la base de datos de Describe que maneja todo tipo de sitios web de donde extrae el texto en codificaciones variadas.

La solución que se implementó consiste en:

a) Tomar el texto en bruto y separarlo por líneas.
b) Analizar si cada línea pertenecía a la codificación UTF-8.
c) En caso afirmativo, se convierte a ISO-8859-1.
d) En caso negativo, se presume que es ISO-8859-1, sin garantía, debido a la falta de herramientas confiables de detección de codificación ISO-8859-1 y que la gran mayoría del texto que no sea UTF-8 es ISO-8859-1.
e) Se une el texto bajo la codificación ISO-8859-1.
f) Se ejecuta POS y la salida se convierte a la codificación a UTF-8 que será la única a lo largo del resto del sistema.

De esta manera se pudieron corregir una gran cantidad de errores de codificación que ocasionaban resultados equívocos. Sin embargo, es apropiado mencionar que el problema de las codificaciones continúa arrojando errores y que la gran cantidad de variables entre ellas, las herramientas disponibles y los errores en los documentos son duros obstáculos que se tienen que superar en el Ecode y en general en el tratamiento de texto en general.

4.2.3 Separador de oraciones

Otro de los problemas de difícil acercamiento en el procesamiento del texto es la distinción entre oraciones; la propia dificultad técnica de encontrar reglas que permitan separar el texto en oraciones se ve opacada por la inconsistencia en la teoría lingüística sobre qué es o no una oración. Para fines prácticos del Ecode, se definió a una oración como una porción de texto entre 2 separadores de oración (. ¿? ¡), aunque existen ciertas reglas antes y después de cada separador de oración que especifica si es o no una oración.

En la siguiente tabla se muestran reglas de separación de oraciones usadas en el Ecode:

Excepción
Abreviaciones: etc.
Títulos: Dr. Ing. Lic. Sr. Sra. Srita.
Marcas de puntuación: …
Reglas de separación
Separador de oración (!?.) espacio
Separador de oración (¡?.) digito

Tabla 4.4 Excepciones y reglas de separación de oraciones

Si bien las reglas y excepciones son muy simples, separan bien el texto para el procesamiento de Ecode; sin embargo, es necesario plantear un análisis más detallado de la conformación de los CDs como oraciones y sobre cómo pueden ser delimitadas para una mayor precisión en este proceso.

En un futuro se debe permitir a los usuarios modificar las reglas de separación de oraciones de acuerdo con sus necesidades; sin embargo, esta posibilidad aún no está implementada.

4.2.4 El etiquetador POS

En un principio Ecode no contaba con un etiquetador POS integrado al sistema. El usuario debía preprocesar su texto para poder ingresarlo al Ecode; de tal manera que fue necesario integrar un etiquetador POS al sistema y permitir libertad al usuario en el procesamiento de sus textos.

El etiquetador POS para el cual el Ecode estaba acoplado era un etiquetador de Brill modificado que funcionaba con un grupo de etiquetas del Corpus del Español Mexicano Contemporáneo (Tabla 2.1, conjunto de etiquetas del CEMC).

Inicialmente, se desarrolló un acoplamiento del etiquetador Brill al Ecode; sin embargo, y aunque funcionaba con buenos resultados, tenía marcadas deficiencias en su desempeño, en velocidad, en tamaño del programa y también en la utilización, por lo que se decidió buscar otro etiquetador POS que permitiera evitar estos problemas y se encontró que una solución mucho más eficiente es el TreeTagger.

El etiquetador TreeTagger provee varias ventajas sobre el etiquetador Brill:

- Es más pequeño.
- Salida fácil de acoplar y por Pipes sin archivos.
- Contiene un conjunto de etiquetas más amplio.
- Permite ser entrenado, aunque ya está entrenado para textos del español.

Considerando estas ventajas se implementó una interfaz entre Ecode y TreeTagger; el principal reto fue que TreeTagger tiene más etiquetas POS que el conjunto del CEMC, por lo que resultó necesario elaborar una tabla de equivalencia y convertir las etiquetas. La tabla de equivalencia es la siguiente:

Etiqueta del CEMC, usadas por Ecode	**Etiquetas de TreeTagger**
/2 (Adjetivo)	ADJ
/1 (Adverbio)	ADV
/6 (Artículo)	ART
/B (Nombre Propio)	NP, ACRNM (Acrónimo), NMON (nombre del mes)
/8 nombre común	NC, NMEA (medida de medición *pesos*, *litros*, etc.)
/4 (preposición)	PREP, PREP/DEL (*del* como preposición)
/3 (Conjunción)	CC, CQUE (*que* como conjunción)
/s (Símbolo)	CM (coma ,), COLON (dos puntos *:*), DOTS (tres puntos ...), LP (paréntesis izquierdo), SEMICOLON (punto y coma), SLASH (diagonal), BACKSLASH (contra diagonal) DASH (guion -), PERCT (Porciento %) QT , comilla (‘), RP (paréntesis derecho) SYM (otros símbolos)
/9-CON (Verbo Conjugado)	VHfin (verbo haber conjugado), VEfin (verbo estar conjugado), VLfin (verbo conjugado), VMfin (verbo modal conjugado), VCLIfin (clítico conjugado), VSfin (verbo ser conjugado)
/9-INF (Verbo en infinitivo)	VEinf (verbo estar en infinitivo), VLinf (verbo en infinitivo), VMinf (verbo modal en infinitivo), VHinf (verbo haber en infinitivo), VCLIinf (clítico en infinitivo), VSinf (verbo ser en infinitivo)
/9-GER (verbo en gerundio)	VEger (verbo estar en gerundio), VLger| (verbo en gerundio), VMger (verbo modal en gerundio), VHger (verbo haber en gerundio), VCLIger (clítico gerundio), VSger (verbo ser en gerundio)
/9-PAR (Verbo en participio)	VEadj (verbo estar en participio), VLadj (verbo en participio), VMadj (verbo modal en participio), VHadj (verbo haber en

	participio), VCLIadj (clítico en participio), VSadj (verbo ser en participio)
/1 (Ambigua, no es usada por Ecode o no se pudo incluir en alguna otra categoría)	ACRNM (acrónimo) ALFP(letras en plural del alfabeto *as, bes*), ALFS (letra del alfabeto), CARD (cardinales), CCAD (conjunción coordinada adversativa *pero*), CCNEG (conjunción coordinada negativa *ni*), CODE(código alfanumérico), CSUBF (conjunción subordinada que introduce clausulas finitas *apenas*), CSUBI (conjunción subordinada que introduce clausulas infinitas *al*), CSUBX (conjunción subordinada de tipo no especificado *aunque*), -DM (pronombres demostrativos, son usados pero interceptados en la gramática POS), FO (formula), FS (marcas de puntuación finales), INT (pronombre interrogativo *quiénes cuántas, cuánto*), ITJN (interjección *oh, ja*), NEG (negación), ORD (ordinal), PAL (*al*), PDEL (*del*), PE (palabra extranjera), PNC (palabra no clasificada), PPC (pronombre personal clítico), PPO(pronombre posesivos *mi, sus, su*), PPX(clíticos y pronombres personales, *nos, me, nosotras, te, sí*), QU (cuantificadores *sendas, cada*), REL (pronombres reflectivos *cuyas, cuyo*), UMMX (unidad de medida *MHz, km, mA*)

Tabla 4.5 Tabla de relación entre etiquetas de etiquetadores

Como se aprecia, muchas etiquetas POS se marcan como ambiguas y esto puede ser por dos razones: que el Ecode no utiliza la etiqueta y por lo tanto la etiqueta fue traducida como ambigua, o bien, que el Ecode sí utiliza ese tipo de palabra, pero que la gramática POS (Archivo 01_gramaticaPOS.pm) detecta la palabra sin importar qué etiqueta se le asignó.

Además, cabe mencionar que dicha relación se realizó sin un análisis profundo de los tipos de palabras que entrega TreeTagger; esto debido a que escapa de la finalidad de este trabajo por lo extenso y laborioso que requiere dicho análisis. Tampoco participó un experto y se realizó sólo por comparación simple entre textos.

Claramente, la relación anterior es susceptible de ser optimizada o corregida por un experto lingüista que permita seleccionar mejores relaciones entre las etiquetas para garantizar que la traducción de palabras sea la adecuada y quizás este pendiente pueda ser parte de un proyecto futuro.

El etiquetador ya presentaba un entrenamiento para textos en español, por lo que no fue necesario reentrenar el TreeTagger; sin embargo, el tipo de lenguaje con el que se definen los CDs es culto, por lo que un reentrenamiento de TreeTagger para textos técnicos y cultos en español permitirá ciertamente una mejora en el etiquetamiento y por lo tanto en el rendimiento general de Ecode.

El sistema se ejecuta enviando el texto a etiquetar por medio de un pipe y se recibe con otro haciendo más eficiente el proceso; posteriormente se procesa la salida del TreeTagger y se vuelven a acoplar las oraciones originales pero ya etiquetadas. En esto consiste el

etiquetamiento de las partes de la oración.

4.3 El módulo de salida

Al ejecutarse el Ecode se obtiene una lista de CDs en un arreglo, por lo que es conveniente procesar más la información para preparar la salida del sistema de acuerdo con las necesidades de los usuarios.

En la siguiente figura se muestra la estructura del módulo de salida:

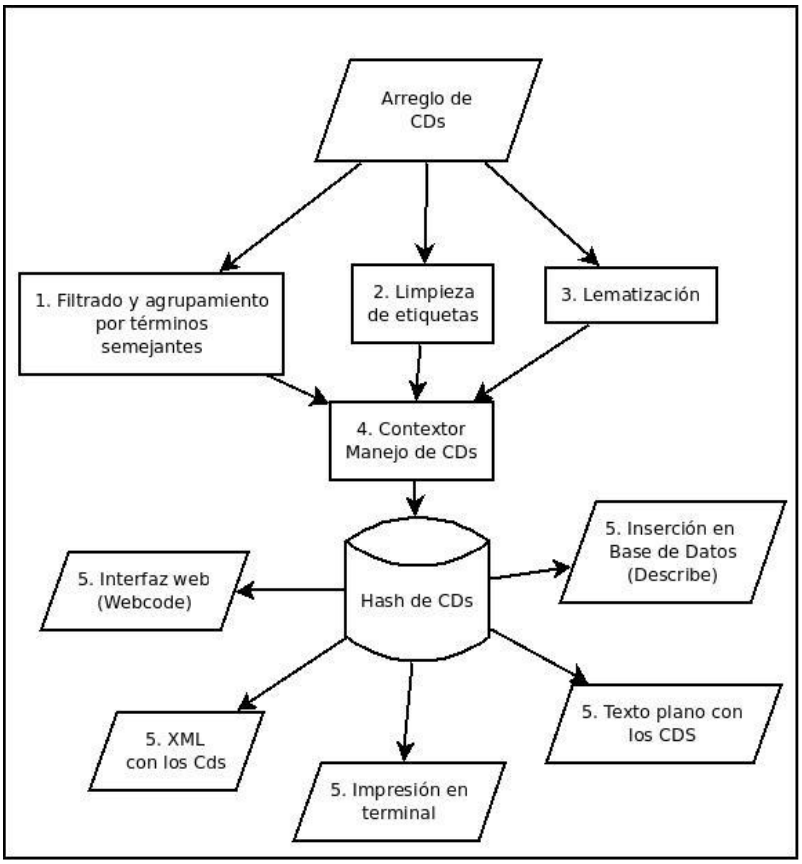

Figura 4.3 El módulo de salida de Ecode

Como se aprecia en la figura, uno de los procesos fundamentales es el número cuatro, "contextor", que tiene la capacidad de extraer los diferentes términos de los CDs y agruparlos por términos semejantes en una estructura de datos que se describirá en breve: el hash de CDs tiene la finalidad de contener a los CDs y presentar una interfaz para las diferentes salidas que requieren los usuarios.

4.3.1. Filtrado y agrupamiento de términos semejantes y asociación de CDs por sus términos

Los CDs, al entrar al módulo de salida en un arreglo, son revisados para extraer el conjunto de términos que corresponde a cada contexto, y posteriormente se aplican reglas que permiten agrupar los términos que son semejantes para luego unificar los CDs que contienen términos parecidos.

La agrupación de términos semejantes, por si sola, requiere de un gran desarrollo lingüístico, por lo que las reglas que se consideraron para agrupar los términos son burdas y únicamente basadas en el contenido de las palabras de los términos y algunas consideraciones de POS. En la siguiente tabla se describen las reglas utilizadas para agrupar y filtrar términos semejantes:

Regla	**Descripción**
Reglas de filtrado	
Tamaño > 10 palabras	Los términos con más de 10 palabras se filtran.
Contiene al término buscado	Si los términos encontrados no contienen al que se ingresó de búsqueda.
Reglas de agrupamiento	
Artículo + término	Se agrupan términos que sean iguales, pero que uno contenga un artículo y el otro no.
Número o símbolo + término	Si el término presenta números o símbolos al inicio, los demás se agrupan.
Palabras funcionales + término	Se agrupan términos que contengan palabras funcionales al inicio o final y que contengan al término semejante.

Tabla 4.6 Reglas de filtrado y agrupamiento de términos semejantes

Un ejemplo de agrupamiento de términos se muestra en la siguiente figura:

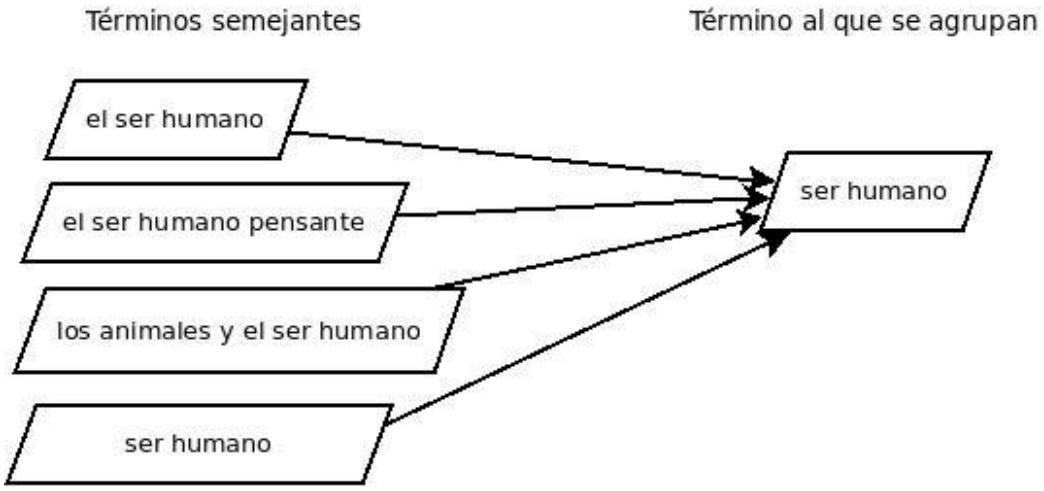

Figura 4.4 Ejemplo de agrupamiento de términos semejantes

Así quedan agrupados los términos semejantes y posteriormente se crea una estructura de datos que contendrá dichos términos agrupados y a los CDs que pertenecen a dicho término.

4.3.2 El hash de CDs

Una vez que se encontraron términos semejantes y se agruparon los contextos definitorios en torno al término que definen, se crea una estructura de datos que permite manipular los CDs y agregarles características, a la vez que se provee una interfaz para la salida de los datos.

La estructura de datos que se forma es un hash asociativo que permite ligar los arreglos de CDs con su término semejante, así como permite el acceso a los CDs que fueron limpiados y/o lematizados.

La estructura del hash de CDs y ejemplos se representan en la siguiente tabla:

Hash de CDs	
Llave: Término semejante	**Valor: Referencia a Hash con arreglo de CDs**
Humano	Puntero a Hash 1
Virus	Puntero a Hash 2
Hash 1	

Llave: CD con etiquetas completas	**Valor: Referencia a Arreglo de limpios-lemas**
<t>el ser humano</t> <pvd lema=ser><vd lema = ser> es </vd> </pvd> <d> un animal</d>	Puntero a arreglo 1
<d> Un ser pensante con un gran cerebro <pvd> es el <vd lema= denominar> denominado</vd> </pvd><t>ser humano</t>	Puntero a arreglo 2
Hash 2	
Llave: CD con etiquetas completas	**Valor: Referencia a Arreglo de limpios-lemas**
<t>un virus <pvd>está <vd lema = denominar> conformado</vd> </pvd><d> por una serie de proteínas.</d>	Puntero a arreglo 3
<d>al ser vivo más pequeño</d><pvd> se le <vd lema = denominar> denomina</vd></pvd> <t> virus.</t>	Puntero a arreglo 4
Arreglo 1	
[0] CD limpio de etiquetas	**[1] CD lematizado**
El ser humano es un animal	El ser hum es un anim
Arreglo 2	
[0] CD limpio de etiquetas	**[1] CD lematizado**
Un ser pensante con un gran cerebro <pvd> es el <vd lema= denominar> denominado</vd> </pvd><t>ser humano</t>	Un ser pens con un gr cerebr es el denom ser hum
Arreglo 3	
[0] CD limpio de etiquetas	**[1] CD lematizado**
Un virus está conformado por una serie de proteínas.	Un vir est conf por una ser de prot
Arreglo 4	
[0] CD limpio de etiquetas	**[1] CD lematizado**
Al ser vivo más pequeño se le denomina virus.	Al ser viv mas peq se le denom vir

Tabla 4.7 Estructura del Hash de CDs

De esta forma se conforma el hash permitiendo al usuario obtener únicamente los CDs cuyo término es semejante al término buscado, o bien una agrupación para su fácil lectura.

4.3.2.1 Limpieza y lematización de CDs

La estructura Hash de CDs al ser creada requiere procesamiento de los CDs para limpiarlos de sus etiquetas y prepararlos para ser usados, ya sea por los usuarios o por otros sistemas que no requieren de las etiquetas de los CDs.

Primeramente, el proceso limpia las etiquetas POS y las etiquetas de partes de los CDs dejándolos como texto limpio, lo cual es más legible para el usuario que sólo se interesa en los CDs y no en sus partes. Posteriormente a los CDs limpios se les aplica un proceso de lematizado que reduce el tamaño de las palabras colocando los lemas de ellas, esto genera que los CDs semejantes sean "medidos" o agrupados con mayor facilidad; además, se permite una salida apta para un mayor análisis lingüístico.

Para lematizar los CDs se utilizó el módulo de Perl "Lingua::Stem::Es" que permite lematizar palabras basado en el algoritmo de Porter.

La lematización se desarrolló en un principio en el Ecode por la necesidad de clasificación de CDs en el módulo consecutivo de Describe, que agrupa los contextos debido a su similitud y hace más adecuado procesarlos estando lematizados.

4.4. Implementación de Ecode

Una vez que se terminaron de construir los módulos de Ecode, fue necesario implementarlos para que los usuarios pudieran trabajar con el sistema; si bien la interacción con las entradas y salidas ya fue completada, era necesario proporcionar al usuario una interfaz con el sistema, además de conectarlo con el sistema Describe.

Las interfaces de usuario son dos: la interfaz web (WebCode) y una interfaz en consola estilo Unix.

4.4.1 Interfaz de usuario web de Ecode (WebCode)

El internet es la vía de transferencia de información más importante. En él se puede enviar todo tipo de datos y accesar aplicaciones Web que permiten una interacción con los usuarios, que pocas veces se obtiene con algún otro tipo de aplicación; esta fue la razón por la que se decidió realizar una interfaz web para Ecode y poder permitir el acceso inmediato de los usuarios.

WebCode se desarrolló en un principio para demostración del funcionamiento de Ecode; sin embargo, ese mostró mucho interés por tener una aplicación web abierta al público.

La aplicación cuenta con todas las características que permite Ecode: tiene una página de entrada en donde el usuario puede subir sus archivos a analizar al servidor o proporcionar una URL a procesar; y una página de salida que permite presentar claramente los CDs encontrados y proporciona herramientas de trabajo, como son el guardar los contextos de interés, obtener algún tipo de CD en específico o presentar los CDs limpios para su lectura o envío de resultados elegidos por correo.

La aplicación web, debido a los requerimientos de Ecode, se montó sobre el servidor que contiene a Describe como una aplicación independiente, aunque por el momento no es accesible para los usuarios pues se encuentra todavía en desarrollo.

4.4.2 Interfaz de consola

El Ecode se desarrolló en un principio como una aplicación de consola que no permitía más argumentos que el archivo de entrada, y no tenía ninguna opción de procesamiento. Se desarrolló, entonces, una interfaz de consola estilo Unix con opciones y argumentos.

La interfaz permite elegir el tipo de entrada y las salidas requeridas, además de contar con ayuda y detectar errores en la sintaxis del programa.

La sintaxis es:

Perl ecode2.pl –opciones Archivo_url_entrada [termino de interés]

En la siguiente tabla se muestran las opciones de la aplicación y su uso:

Opción	Descripción
Entradas:	
-a	La entrada es un archivo (txt, pdf, html, xml, doc)
-u -	La entrada es una URL (debe contener "http://" al inicio)
-i	Busca un término en específico que tiene ser el segundo argumento después de la entrada
Salidas:	
-e	Quita etiquetas de partes de CDs (incluye etiquetas de paso de información)
-p	Quita las etiquetas POS
-l	Archivo de CDs limpios sin POS y sin etiquetas de partes de CDs (CDs_limpios.txt)
-m	CDs con su texto lematizado (CDs_lematizados.txt)
-x	Todas las salidas: principal con etiquetas (depende de -e y -p); CDs limpios; CDs lematizados.
-t	Archivo de texto plano de entrada extraído de la fuente y separado por oraciones (texto_extraido.txt)
-g	Archivo con el texto de entrada y etiquetado con POS
Otros:	
-d	Debug (genera archivo de debuggeo)
-h	Uso del sistema
-s	Modo silencioso
Del formato de Salida: La salida es uno o varios archivos en el directorio /sal : + (CDs_final.txt) - Archivo conteniendo los CDs según fueron requeridos en la entrada + (CDs_lematizados.txt) - Archivo que contiene a los CDs lematizados [opcional] + (CDs_limpios.txt) - Archivo que contiene a los CDs limpios [opcional]	

+ (Texto_Extraido.txt) - Texto extraído de la fuente original [opcional] + (Texto_Extraido_POS.txt) - Texto extraído y etiquetado con POS [opcional]
Ejemplos: + Busca CDs en archivo.pdf (-a), limpio de etiquetas de partes de CDs (-e), sin etiquetas POS (-p), con archivo de texto extraído intermedio: **perl ecode2.pl -e -p -t -a archivo.pdf** + Busca CDs en URL http://es.wikipedia.org/wiki/Ling%C3%BC%C3%ADstica_computacional (-u) y busca el término "lingüística computacional" (-i) perl ecode2.pl -u –i “http://es.wikipedia.org/wiki/Ling%C3%BC%C3%ADstica_computacional" "lingüística computacional"

Tabla 4.8 Argumentos y uso de la interfaz de consola de Ecode

4.4.3 Integración a Describe

Si bien se pensaba que el sistema Ecode sería el motor de extracción de información del proyecto Describe, el sistema requería desarrollo para los requerimientos de Describe, por lo que ambos sistemas estaban desconectados y se tuvo que diseñar una interfaz que permitiera conectar a Ecode a las necesidades del Describe.

Para acoplar el Ecode al sistema Describe fue necesario hacer un análisis de las posibles formas de conectar los demás módulos de Describe escritos en Java con el Ecode escrito en Perl. Entonces, se consideró pasar la información en archivos y hacerlo por tuberías (*Pipes*). Para poder llevar a cabo un control más estricto con estadísticas en Describe, se decidió que fueran módulos independientes y se comunicaran a través de una base de datos.

Esta base de datos es un conjunto de tablas que permite mantener el control sobre los términos buscados, los documentos de dichos términos, el estado del procesamiento del término, los CDs de Ecode, además de los campos y tablas para los diferentes módulos, tanto funcionales como experimentales que lo componen.

La base de datos de Describe está montada con Mysql, por lo que se tuvo que utilizar el módulo estándar de Perl, DBI, que permite conectar a Perl con cualquier base de datos incluyendo a Mysql. El módulo que maneja la entrada y salida de Ecode con la base de datos de Describe se escribió en un archivo aparte (Archivo: ES_DB_describe.pm).

En la siguiente figura se muestra como interactúa Ecode con la base de datos de Describe:

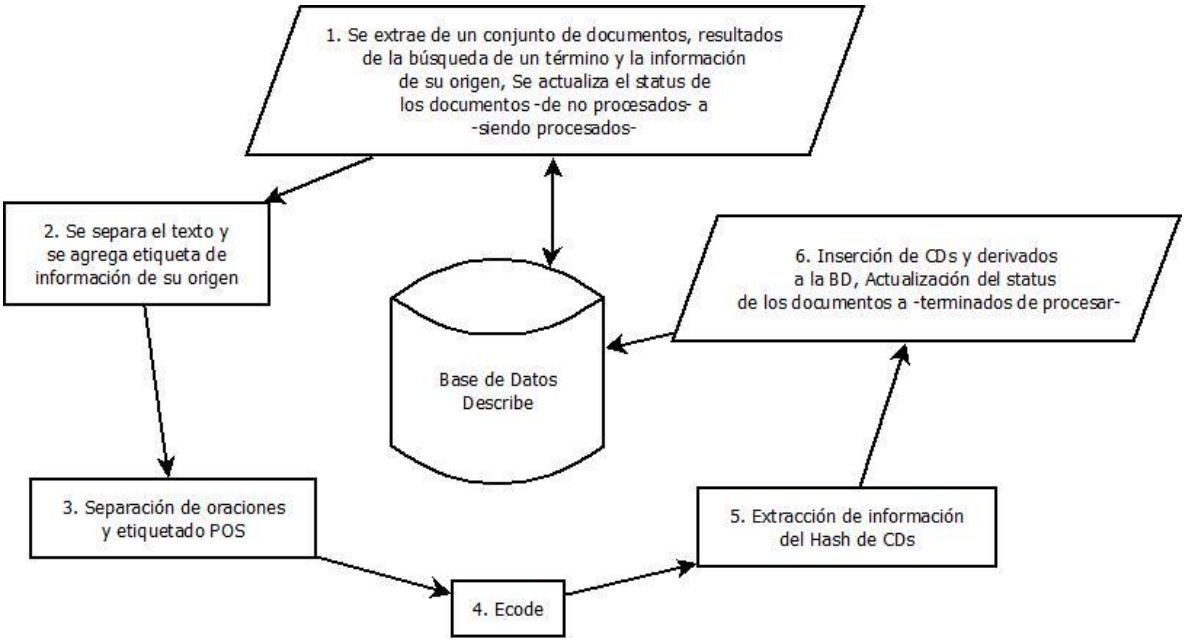

Figura 4.5 Interacción de Ecode con la base de datos de Describe

En la figura, el número uno representa la alimentación de Ecode con texto, que es una consulta para buscar qué documentos de un término o términos ya están listos para ser procesados. En esta consulta, además, se trae información del origen de los documentos, URL, término buscado, id de documento, etc. Además se inserta información en una tabla, indicando que los documentos serán procesados por Ecode con el tiempo, para llevar estadísticas y cálculo de resultados.

El número dos es sólo un preprocesamiento que permite añadir a cada oración una etiqueta que no será tocada por Ecode durante su procesamiento y permite pasar información sobre el origen de la línea, lo que finalmente permitirá saber de dónde se obtuvo un CDs.

Los números tres, cuatro y cinco corresponden en sí al Ecode: el tres es el módulo de entrada, el cuatro es el procesamiento principal y el cinco es la preparación de los CDs para sus diferentes salidas, es decir, es el módulo de salida.

La inserción final a la base de datos está dada por el número seis; en este paso se insertan los CDs, los CDs lematizados y los CDs limpios a la base de datos, decodificando la etiqueta de información e insertando los datos en la tabla correspondiente. Además, se marca el estado de los documentos procesados como terminados para que puedan ser atendidos por el módulo siguiente.

5. Resultados y pruebas

El Ecode, al ser un sistema de gran tamaño y con un grado de complejidad elevado, presenta una gran cantidad de parámetros a evaluar. Debido a que el alcance de este trabajo es limitado se decidió tomar las medidas más significativas y de más fácil medición para evaluar el sistema.

Si bien éstas no son ni remotamente las únicas medidas y un trabajo de tal complejidad requeriría de una evaluación mucho más exhaustiva y considerar muchos más parámetros, se acotó la evaluación a parámetros cuyo cambio resultó de un gran impacto para el sistema: el tiempo de ejecución tanto de los etiquetadores POS como del sistema global, simplificación y reducción de líneas de código y la precisión y exhaustividad del sistema previo y el producto medido con un corpus de pruebas.

5.1 Simplificación de código

Al realizar la optimización se redujeron algunas líneas de código, lo cual impacta directamente la legibilidad y desarrollo futuro del código e, indirectamente, en el tiempo de ejecución del sistema, rendimiento analizado en otro apartado.

Parte de la simplificación del código consiste en reducir las operaciones de entrada y salida, lo cual también es medido y se presentan los resultados en cuánto se redujeron.

Para la compresión de código se midió el número de líneas de cada archivo del Ecode original y sus correspondientes en el producto final, lo que está estrechamente relacionado con el tiempo de ejecución.

En el capítulo anterior se planteó como sería compactado el código, sus optimizaciones y la reescritura de expresiones regulares. Después de haber realizado las mediciones a los archivos se obtuvieron resultados favorables, sin embargo, cabe mencionar las siguientes consideraciones.

Las diferentes acciones de optimización, que fueron planteadas por separado, muchas veces van compaginadas unas con otras, ya que en el código no es posible o no es conveniente tratarlas por separado. Por esta razón se decidió medir directamente la cantidad de líneas afectadas por cada archivo y no analizar por separado los diferentes tipos de optimizaciones.

Esta medición es relativa y lo único que indica es que se redujo la cantidad de líneas y que

el código es ahora más compacto, el impacto de estos cambios va estrechamente relacionado con el tiempo de ejecución.

5.1.1 Reducción de operaciones de entrada y salida

Un paso esencial de la simplificación del código fue la reducción al máximo de operaciones de entrada y salida; en el capítulo anterior se analizaron las lecturas y escritura al disco duro durante los módulos, en la Tabla 4.2 se muestra cuántas lecturas y escrituras de archivos se producen en el sistema.

Archivo (Módulo)	**Operaciones de Entrada (lectura de archivos)**	**Operaciones de Salida (Escritura de archivos)**
01_vds.pm	1	2
02_pvds.pm	2	2
03_td.pm	2	1
04_filtro.pm	3	2
05_arbol.pm	2	2
06_retagging.pm	1	1
07_rankingTyD.pm	2	1
08_rankingGlobal.pm	2	1
09_final.pm	7	6
01_gramaticaPOS.pm	0	0
ecode.pl	1	3

Tabla 4.2[15] Ocurrencias de operaciones de Entrada/Salida en los módulos de Ecode.

Luego de eliminar todas las entradas y salidas innecesarias al sistema, se obtuvo el siguiente resultado:

[15] Tabla del capítulo 4

Archivo (Módulo)	**Operaciones de Entrada (lectura de archivos)**	**Operaciones de Salida (Escritura de archivos)**
01_vds.pm	0	0
02_pvds.pm	0	0
03_td.pm	0	0
04_filtro.pm	0	0
05_arbol.pm	0	0
06_retagging.pm	0	0
07_rankingTyD.pm	0	0
08_rankingGlobal.pm	0	0
09_final.pm	0	0
01_gramaticaPOS.pm	0	0
ecode.pl	1	2
00_Inicia.pm	2	0
Módulo de entrada	Depende del usuario (usualmente 1)	0
Módulo de salida	0	Depende del usuario (usualmente 1)

Tabla 5.1 Ocurrencias de operaciones de Entrada/Salida en los módulos de Ecode después de su optimización

De esta manera el núcleo del sistema queda libre de lectura o escritura al disco duro, lo que genera un Ecode más rápido que sólo se ejecuta en la memoria.

5.1.2 Compresión del código

Para simplificar el código se tomaron en cuenta los cinco puntos planteados en el capítulo 4: inicialización de variables, simplificación de variables y su alcance, reescritura de expresiones regulares, compresión de estructuras de control y conjunción de ciclos similares.

El resultado final es que la compactación de líneas de código y simplificación, en general fue considerable, lo que va respaldado con los tiempos de ejecución encontrados. Para la reducción de líneas la siguiente tabla muestra el número de líneas por archivo del Ecode original:

Archivo (Módulo)	**Número de líneas Del Ecode original**	**Número de líneas en el producto final**	**% de reducción** (de acuerdo al original)
01_vds.pm	178	97	54.5
02_pvds.pm	287	195	67.94
03_td.pm	295	212	71.86
04_filtro.pm	301	241	80
05_arbol.pm	824	645	78.28
06_retagging.pm	218	240	110
07_rankingTyD.pm	394	327	83
08_rankingGlobal.pm	160	112	70
09_final.pm	125	57	45.6
01_gramaticaPOS.pm	98	92	93.88
00_Inicia.pm	Inexistente	100	130[16]
Global:	**2880**	**2318**	**80.49**

Tabla 5.2 Número de líneas de cada archivo del Ecode original y el final

[16] El 130 en realidad es ficticio y solo se agrega para obtener un valor que complete para el 80.49, en realidad este valor se refiere al porcentaje de las líneas que se sacaron de otros archivos y se agregaron 00_Inicia.pm

Al final, encontramos que la reducción global de líneas de código fue un 80.5% respecto del original.

5.2 Tiempo de ejecución

Se consideraron dos mediciones: primero, al realizar el módulo de entrada se sustituyó el programa de etiquetado POS basado en el algoritmo de Brill por el TreeTagger, el cual es mucho más rápido y es precisamente una de las mediciones a realizar: el tiempo de ejecución de los etiquetadores POS; y segundo, se midió únicamente el módulo principal de procesamiento, es decir, lo que era el Ecode original y el resultado de la optimización, para ver cuánto se logro reducir el tiempo que tarda en extraer los CDs.

Como corpus de prueba se utilizó un archivo con alrededor de 5 mil líneas que, a pesar de no ser muy grande, sí proporciona un tamaño adecuado para discernir las diferencias entre un Ecode y el otro.

5.2.1 Tiempo de ejecución de etiquetadores POS

Para la medición del tiempo de ejecución de los etiquetadores POS se utilizó un corpus (Archivo: documentos_medicion.txt) obtenido de diferentes textos elegidos al azar de la base de datos de Describe y cuenta con un total de 5297 líneas de tamaño variable, 366 *Kbytes,* que tras el filtrado del preprocesamiento sólo son validas y se procesan 4248 líneas.

Se realizó un preprocesamiento al texto antes de ambos etiquetadores, el que sería requerido para el etiquetado POS de Ecode, que principalmente consta de eliminación de líneas vacías o muy pequeñas y separación en oraciones. El tiempo de ejecución del preprocesamiento está considerado en ambas mediciones, en parte para asegurar que ambos etiquetadores recibieran la misma información y que si existe un error en el corpus de entrada, sea corregido y no intervenga en los etiquetadores.

Los resultados obtenidos se muestran en la siguiente tabla:

Corrida	Etiquetador Brill [s]	TreeTagger [s]	% tiempo que tarda el TreeTagger vs Brill
1	68.08	3.37	4.95005875
2	67.69	3.39	5.00812528
3	66.60	3.38	5.07507508
PROMEDIO	**67.46**	**3.38**	**5.0106241**

Tabla 5.3 Comparación de tiempos de ejecución de etiquetadores POS

En esta tabla, claramente se aprecia la reducción sustancial del tiempo de etiquetado POS con una reducción de 67.5 [s] a 3.38 [s], lo que se traduce en realizar el trabajo en sólo el 5% del tiempo que le tomaba originalmente.

5.2.2 Tiempo de ejecución del núcleo de Ecode

Para medir los tiempos de ejecución del procesamiento principal del Ecode y por lo tanto el impacto de la optimización que se le hizo al código, se utilizó el mismo corpus del apartado anterior (Archivo: documentos_medicion.txt), pero previamente etiquetado con POS, usando TreeTagger; sin embargo, debido a que ambos sistemas son muy rápidos y para poder tener una medida más precisa de la diferencia, se ejecutan 10 veces por cada medición.

Los resultados para el Ecode original y el producto final se muestran en la siguiente tabla:

Corrida (X 10)	Ecode original [s]	Ecode optimizado [s]	% optimización de tiempo
1	13.05	12.21	93.56
2	13.17	12.08	91.72
3	13.04	12.03	91.25
PROMEDIO	13.09	12.11	92.51

Tabla 5.4 Comparación de tiempos de ejecución del Ecode original contra el producto de la optimización.

De este modo se comprueba que se logro reducir casi en 10% el tiempo que tarda uno y otro sistema en su procesamiento principal.

5.3 Precisión y exhaustividad de los contextos extraídos contra el Ecode original

Para medir éstos parámetros se utilizó un módulo de evaluación (Archivo 10_eva.pm) del Dr. Alarcón. Asimismo, construí un corpus de 400 candidatos a contextos definitorios (Archivo: CORPUS.txt), tomando líneas al azar del corpus de trabajo del Dr. Alarcón (Alarcón 2009). Este corpus fue analizado para determinar si son o no CDs, y etiquetado por un experto, el Lic. Víctor Mijangos, que tiene conocimiento de la estructura de los contextos (Mijangos 2011) y una formación en lingüística. Con esto fue posible medir la precisión y exhaustividad de ambos sistemas.

De acuerdo con Jurafsky y Martin (2000), la precisión es una medida que se utiliza para determinar cuánta información extraída automáticamente por el sistema es correcta, mientras que la cobertura es una medida para saber cuánta de la información relevante en el texto fue extraída automáticamente.

En Alarcón (2009) se definen las fórmulas de la obtención de la precisión y exhaustividad del sistema:

$$\text{Precisión} = \frac{\text{número de respuestas válidas propuetas por el sistema}}{\text{número de respuestas propuestas por el sistema}}$$

$$\text{Cobertura} = \frac{\text{número de respuestas válidas propuetas por el sistema}}{\text{número total de respuestas en el texto}}$$

Ahora bien, pensando en el escenario de la extracción de CDs, estas fórmulas las podemos interpretar de la siguiente manera:

$$\text{Precisión} = \frac{\text{número de CDs válidos propuetos por el sistema}}{\text{número de CDs propuestos por el sistema}}$$

$$\text{Cobertura} = \frac{\text{número de CDs válidos propuetos por el sistema}}{\text{número total de CDs en el corpus}}$$

Figura 5.2 Fórmulas de precisión y exhaustividad (Alarcón 2009: 202)

Además, se adaptó el módulo de evaluación (Archivo: 10_eva.pm), el cual analiza la precisión y exhaustividad de un corpus etiquetado.

Los resultados son:

Sistema	CDs en el corpus (experto)	CDs propuestos por el sistema (automáticos)	CDs válidos propuestos por el sistema
Ecode original	150	305	126
Ecode Final	150	344	134

Tabla 5.5 Resultados globales por CDs

En números, calculando la precisión y exhaustividad:

Medición	Ecode original	Ecode Final
Precisión	0.4131147	0.3895348
Exhaustividad	0.8235394	0.8815789

Tabla 5.6 Resultados de precisión y exhaustividad

Estos resultados muestran que el Ecode original es un poco más preciso, pero el Ecode final resulta tener mayor cobertura de contextos definitorios.

5.4 Recapitulación del trabajo

De los resultados consideré varios puntos que vale la pena mencionar:

Para la simplificación del código, en cuanto a la compresión de código, se redujo en 20% el número de líneas, sin embargo, más que en la cantidad se obtuvo una optimización que permite tener código bien delimitado, modularizado y más legible para que pueda ser mejorado posteriormente. Además se redujeron prácticamente todas las escrituras Entrada y Salida a disco duro excepto en la entrada y la salida de Ecode, es decir, todo lo trabaja en memoria.

En cuanto al tiempo de ejecución, la medición se realizó con un corpus muy pequeño. Para

un corpus de un tamaño más considerable, debido a las operaciones de entrada y salida, el Ecode Final terminaría primero con una diferencia más significativa.

El cambio del etiquetador POS logró reducir 20 veces el tiempo de etiquetado, además de que TreeTagger es mucho más fácil de configurar para calibrarlo con textos afines y que sea un poco más preciso.

En el núcleo de Ecode se redujo el tiempo de ejecución en 5%, lo que si bien es poco, en corpus de gran tamaño este valor es significativo.

La precisión y exhaustividad son medidas de las cuales es importante mencionar que:

- Si bien se realizó toda una reestructura del código, la lógica del núcleo de Ecode no fue cambiada por no poseer los conocimientos lingüísticos como para tal empresa; por lo que la mayor parte de la diferencia entre los valores de los sistemas se debe a los etiquetadores POS, de nuevo este valor se puede mejorar mucho entrenando a TreeTagger para textos similares a los corpus de contextos definitorios.
- El experto tomó en cuenta todos los niveles lingüísticos para filtrar los candidatos del corpus de entrada, no solo los patrones morfo-sintácticos, sino la semántica de las palabras que no son considerados por Ecode; esto da una gran diferencia entre los contextos del experto (150) y los del Ecode original y final, 305 y 344 respectivamente. Así pues, aunque exista esta diferencia, también difieren los 2 sistemas, por lo que es posible medir qué tan precisos son los sistemas.

Además, cabe mencionar que los valores de precisión (0.53) y cobertura (0.79) plasmados en Alarcón (2009:203) difieren mucho a los obtenidos en este trabajo, tanto en el Ecode original como en el final debido a dos factores: al corpus utilizado y al experto que los clasificó. Si bien el corpus que aquí se utilizó fue obtenido de un corpus del propio Alarcón, no fue de su corpus final de pruebas, ya que no se tuvo acceso a él, pero se obtuvo de uno de sus corpus intermedios.

6. Conclusiones

Durante este trabajo he desarrollado una herramienta flexible y fácil de usar a partir de Ecode, logré optimizar el sistema original y lo expandí para abarcar un poco más del análisis de los contextos definitorios. Esta herramienta servirá como base para futuros sistemas similares.

6.1 Recapitulación del trabajo

Durante la realización de esta tesis tuve que aprender teoría lingüística y de PLN y aplicar mis habilidades como desarrollador y programador en la práctica.

Estudié Ecode en su totalidad y pude encontrar las necesidades inmediatas del sistema y resolverlas. Busqué parámetros adecuados para su medición, además de permitir que los usuarios extrajeran contextos definitorios durante todo el desarrollo del presente sistema. Se lograron los objetivos de la tesis. Las conclusiones se exponen a continuación.

6.2 Conclusiones

Las conclusiones, al igual que el desarrollo del trabajo, las divido en tres partes, a saber, la optimización, la expansión y finalmente la adaptación.

6.2.1 Conclusiones sobre la optimización

La optimización fue significativa en cuanto al núcleo del Ecode: se redujo la lectura y escritura al disco duro, de modo tal que únicamente el acceso al disco es en los módulos de entrada y de salida, lo que repercute de manera directa en el desempeño del sistema.

Se redujo el código a un 80%, lo cual impacta directamente en el tiempo de ejecución del sistema, además de presentar un código más legible, más organizado y más fácil de modificar en futuros desarrollos.

De la reducción de operaciones de entrada y salida, y de la reducción del código se desprende directamente el tiempo de ejecución, que es una medición que muestra el impacto de los puntos anteriores en el sistema. Para el núcleo del Ecode se logró reducir el tiempo de ejecución en aproximadamente 10%, lo cual quizá no parezca significativo, pero sí lo fue para Ecode, pues se tiene que tomar en cuenta que el tiempo de ejecución crece de manera exponencial con el tamaño de los datos.

Se redujo el tiempo que tarda el sistema en etiquetar las palabras con su categoría gramatical en un 95%; además, este proceso no estaba integrado al sistema antes y era necesario llevarlo a cabo.

En la medición de la precisión y la exhaustividad influyen factores propios de la expansión, sin embargo, cabe mencionar que después de la optimización y de agregar los módulos de entrada y salida de la expansión, la precisión se redujo de 41% a 39%, y la cobertura aumentó de 82% a 88%.

6.2.2 Sobre la expansión

La expansión consistió en añadir un módulo de entrada y uno de salida que permitiera un abanico de opciones de entrada o de salida de acuerdo con las distintas necesidades.

En el módulo de entrada se agregó un separador de oraciones, se integró el etiquetador POS TreeTagger y se permitió el paso de información durante el sistema por medio de etiquetas especiales al inicio de las líneas de texto.

El módulo de salida consiste en la presentación de los CDs con sus diferentes conjuntos de etiquetas (POS, de partes del CD y especiales), la agrupación por términos semejantes y la búsqueda de un solo término en la salida.

6.2.3 Sobre la adaptación

Adaptar el sistema consistió en condicionarlo para que además de funcionar como una herramienta individual, funcione como una librería que pueda ser incluida y utilizada en futuros proyectos, sin necesidad de reescribir o proveer una futura adaptación; esta librería es el núcleo del sistema y está escrita en Perl.

Otro punto de la adaptación fue permitir a Ecode interactuar con la base de datos de Describe –y en general puede interactuar con cualquier base de datos con una ligera adaptación–. Para esto ya está programada la conexión con las bases de datos, tanto para la obtención de los datos como para insertar los contextos de salida con los datos pertinentes a estos, el origen, término buscado, línea en la que aparece, etc.

Estas adaptaciones a Ecode ya se encuentran plenamente funcionales y listas para su aplicación a futuros proyectos.

6.3 Trabajo futuro

El Ecode, si bien es una herramienta completa e integral, requiere de mucho desarrollo para abarcar las necesidades que no fueron planeadas pero fueron surgiendo durante la realización del sistema.

El trabajo futuro consiste en varias acciones tanto de carácter lingüístico como ingenieril y las describo a continuación:

6.3.1 Trabajo futuro sobre los módulos de entrada y salida

Los módulos de entrada y salida requieren de mayor desarrollo para tener un sistema que permita obtener y entregar los resultados aún con más flexibilidad que la que actualmente se tiene.

El módulo de entrada requiere una librería o programa que permita extraer el texto de los PDFs con mayor precisión y para diferentes versiones de este formato de archivo. Requiere además extracción de texto en formato de Open Office (odt) y también de otros formatos de archivo relevantes (DjVu, PowerPoint, Ghost Script, Latex, etc.).

Además se requiere explorar, con la ayuda de un lingüista, las separaciones de las oraciones dentro de los archivos, ya que no siempre son separadas de manera adecuada, ya sea porque vienen mal separadas de origen o por el separador de oraciones implementado.

El etiquetador podría mejorar en gran medida los resultados de Ecode si fuera entrenado con un texto especializado que contenga contextos definitorios y si se contara con un experto que pueda indicar las palabras que son etiquetadas de manera correcta y las que no. Además, algunas palabras funcionales, que pueden tener diferente categoría gramatical según su contexto, frecuentemente son confundidas por el etiquetador y debería agregarse la opción de corregir dichas palabras dentro del módulo de entrada.

El problema más complicado que requiere de solución es el manejo de las codificaciones de caracteres, debido a que las múltiples entradas de datos por lo regular manejan codificaciones diferentes, son difíciles de controlar y las herramientas disponibles son ineficientes. Este sistema tiene un módulo para el manejo de las codificaciones, pero se requiere de una investigación sobre los tipos de codificaciones posibles en las entradas y cómo distinguir cada una de ellas, ya que algunas veces es imposible determinarlo.

En el módulo de salida es necesario mejorar el agrupador de términos semejantes con la ayuda de un lingüista, ya que esta tarea es complicada y requiere de un análisis para determinar en cuáles circunstancias es conveniente agrupar los términos. De manera similar, es necesario profundizar en el filtrado de términos erróneos o mal formados que suelen aparecer en la salida.

Se necesita implementar que la salida sea en XML, crear una hoja de estilo para que la presentación sea más agradable. Además se pretende que haya una estandarización en las etiquetas de los CDs, tanto en el CORCODE como en los otros proyectos existentes dentro del GIL y en los proyectos futuros.

Otro requerimiento es permitir la configuración de los accesos a bases de datos en un archivo o en los argumentos que recibe al ejecutarse. Por ahora es estático y la base de datos de Describe cambia su estructura regularmente, por ello la única forma de configurarlo para los cambios es en el código.

6.3.2 Trabajo futuro sobre el núcleo del sistema

El núcleo del sistema requiere de varios cambios para hacerlo configurable en sus parámetros y que por lo tanto pueda ser acondicionado a los requerimientos de futuras investigaciones.

Se debe implementar una interfaz de configuración con las gramáticas ya existentes. Buscar como agregar a las gramáticas parámetros que aun no son configurables, como la escritura de etiquetas POS, agregar nuevos filtros, etc.

Si bien el sistema cubre una gran cantidad de casos en la identificación de los contextos definitorios, aún queda mucho por investigar. Además deberían agregarse al Ecode los recientes avances en la teoría lingüística sobre contextos definitorios.

7. Bibliografía

- Alarcón, R. (2009), *Descripción y evaluación de un sistema basado en reglas para la extracción automática de contextos definitorios*. Tesis doctoral. Universitat Pompeu Fabra. Barcelona.

- Alarcón, R. Bach, C. y Sierra G. (2008), "Extracción de contextos definitorios en corpus especializados: Hacia una elaboración de una herramienta de ayuda terminográfica", Revista Española de Lingüística, vol. 37, 2008, pp. 247-27. Barcelona.

- Cabré, M. T. (1993), *La terminología. Teoría, metodología, aplicaciones*. Editorial Antártica / Empúries. Barcelona.

- De Bessé, B. (1991), "Le ContexteTerminographique". Meta 26(1):111-120. Francia.

- Forney, D. (1973), "The Viterbi Algorithm". Institute of Electrical and Electronics Engineers vol.61 no. 3. USA.

- Jurafsky, D. y Martin, J. (2006), *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Education. USA.

- Leech, G. y Wilson, A. (1999), *Recommendations for the Morphosyntactic Annotation of Corpora*. EAGLES Report EAG-TCWG-MAC/R. USA.

- Méndez, C. (2009), *Identificación Automática de categorías gramaticales en español del siglo XVI*. Tesis de Maestría. Universidad Nacional Autónoma de México. México.

- Pearson, J. (1998), *Terms in Context*, John Benjamins Publishing Company, Amsterdam.

- Sager, J. C. (1993), *Curso práctico sobre el procesamiento de la terminología.* Pirámide. Madrid. [Título Original: A Practical Course on Terminology Processing, 1990; traducción de Laura Chumillas Moya]

- Schmid, H. (1994), “Probabilistic Part-of-Speech Tagging Using Decision Trees”, Conferencia Internacional New Methods in Language Processing. UK.

- Sierra, G. (2009), “Extracción de contextos definitorios en textos de especialidad a partir del reconocimiento de patrones lingüísticos”. *Linguamática*, Num 2, 13-37. Portugal.

- Sierra, G., Alarcón, R., Molina, A. y Aldana, E. (2009), “Web Exploitation for Definition Extraction”. Latin American Web Congress. México.

- Sierra, G. y Alarcón, R. (2010), *A Rule-based Algorithm for Lexical Knowledge Extraction*. Sin publicar. México.

- Sierra, G. y Alarcón, R. (2010), “Minería de Textos para la Búsqueda de Información Especializada”, en *Komputer Sapiens*, Año II Vol. I, 23-27. Mexico.