Proyecto de Grado

Técnicas de estado finito para la extracción automática de metadata en la Web

Tutores:

MSc. Ing. Regina Motz
Ing. Guillermo Moncecchi

Integrantes:

Marcia Porteiro
Rodolfo Paiva

Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Año 2003 - 2004

RESUMEN

Cada vez más, la información se presenta como la principal fuente de riqueza. Es en este sentido que Internet brinda gran potencia, permitiendo el acceso al mayor repositorio de datos existente. En este contexto, los mecanismos de acceso a dicha información se vuelven críticos a la hora de intentar maximizar el aprovechamiento y disponibilidad de la misma.

La falta de estructuración de la información electrónicamente disponible, hace difícil las tareas de manipulación de los datos involucrados, imposibilitando el uso de técnicas tradicionales de consultas de base de datos. Muchos de estos documentos, sin embargo, contienen abundantes elementos que en su conjunto describen la esencia del contenido del documento, cuyo aprovechamiento requiere de varios factores.

Uno de estos factores es la existencia de información descriptiva del contenido semántico de los documentos. Este tipo de información es en la actualidad escasa, y de mala calidad.

Se han hecho diversos intentos con el fin de facilitar la generación y mejorar la calidad de dicha información descriptiva.

En este proyecto, se busca atacar el problema mencionado desde un enfoque novedoso, el cual se basa en la utilización Técnicas de Estado Finito como medio para permitir la automatización del proceso de generación de la información mencionada.

ÍNDICE

Capítulo Nº 1: INTRODUCCIÓN

El presente proyecto titulado “Técnicas de estado finito para la extracción automática de metadata en la Web” se enmarca dentro de la asignatura Proyecto de Grado del Instituto de Computación de la Facultad de Ingeniería. El mismo tiene una estrecha vinculación con dos de las áreas de mayor interés en lo que a computación respecta: el Área de Concepción en Sistemas de Información y el Área de Procesamiento de Lenguaje Natural.

El proyecto consistió principalmente en el estudio de los factores involucrados en la extracción automática de la metadata asociada a una página Web dada, utilizando herramientas de estado finito, tales como transductores, para este fin. El análisis se materializa con el desarrollo incremental de un prototipo, que refleja las principales características detectadas en pro de realizar un procesamiento eficaz de la información.

El crecimiento del uso de metadatas brindará un amplio espectro de posibilidades a los usuarios a la hora de ubicar y entender las principales características de los datos. A pesar de los beneficios de su utilización, actualmente los documentos localizados en la Web no se encuentran enriquecidos por lo general con esta información, lo que hace más difícil la tarea de manipulación y mantenimiento del vasto flujo de información que representan.

Si bien la idea de generar automáticamente la metadata asociada a documentos presentes en un ambiente Web no es nueva, los enfoques propuestos se ven limitados debido a la inherente vinculación que ofrecen con la estructura de los documentos. La propuesta del uso de ontologías combinadas con máquinas de estado finito presenta una nueva visión para la generación.

Este informe pretende delinear los rasgos más sobresalientes relevados a lo largo de la duración del proyecto, incluyendo los principales problemas encontrados a la hora de desarrollar un sistema de estas características, evaluando en cada caso las posibles soluciones.

1.1 Motivación y objetivos

La Web se ha convertido en el principal y más accedido repositorio de información global, ya que comprende millones de documentos que son accedidos por millones de usuarios abarcando muchos aspectos de la vida cotidiana, laboral o empresarial.

El considerable volumen y heterogeneidad de la información existente en este escenario, hace de interés el abordaje de técnicas o metodologías para su manipulación y recuperación. Cabe destacar que la mayoría de las técnicas de búsqueda de información que son utilizadas actualmente en este medio, tienen deficiencias en cuanto al contenido semántico de los documentos que se recuperan, ya que muchas veces estos no pertenecen al dominio de información deseado, y es el usuario el encargado de realizar manualmente un refinamiento en los resultados de una búsqueda para descartar documentos que no son de interés.

Con el fin de subsanar este problema es que aparece la idea del procesamiento semántico de los documentos (idea básica de lo que se conoce como la Web Semántica [MOU01, VEN, WS]), es decir ser capaces de tener agentes de software con la inteligencia suficiente para procesar automáticamente los documentos, realizando un análisis semántico de su contenido.

Para hacer posible esto no basta solamente con la existencia de dichos agentes [FRA96, HEN99], son también necesarios otros dos elementos. El primero es la existencia de una ontología del dominio en el que estemos trabajando. Esto es ni más ni menos que una conceptualización de un dominio común, es decir una forma de estandarizar la manera en que denominamos a los diferentes conceptos de nuestro dominio. El segundo concepto, que es el que más interesa a los efectos de nuestro proyecto, es la existencia de información asociada a los documentos que describa el contenido semántico de los mismos. Para hacer referencia a datos que describen de alguna forma u otra, el contenido semántico del documento al que están asociados utilizaremos el término metadata.

Esta metadata descriptiva de los documentos, si bien es esencial para permitir el análisis semántico de los mismos, en la actualidad por lo general no está presente, y si lo está probablemente sea muy pobre. Es en esta dirección que apunta nuestro proyecto, buscando estudiar un mecanismo para la generación de esta metadata, que permita enriquecer los documentos haciendo más fácil su procesamiento desde el punto de vista semántico.

El presente proyecto tiene como objetivo principal el estudio de la viabilidad de extraer automáticamente la metadata asociada a páginas Web relevantes para un dominio específico, utilizando para este fin, herramientas de estado finito tales como transductores. Para realizar dicha evaluación se construirá un prototipo de manera iterativa e incremental, el cual intentará, partiendo de un documento sin etiquetar y la especificación de la ontología asociada al dominio del documento, enriquecer dicho documento agregándole la información semántica correspondiente, utilizando técnicas de procesamiento de lenguaje natural.

1.2 Alcance del Proyecto

Como se mencionaba en el punto anterior, el objetivo de este proyecto es evaluar una técnica para lograr un determinado propósito. Por este motivo, el trabajo presentado tiene una fuerte componente en lo que refiere a investigación de técnicas y herramientas existentes relacionadas con el dominio del problema, siendo parte importante del proyecto la documentación asociada a la misma. Asimismo, los resultados de dicha investigación consolidarán uno de los pilares básicos para la descripción de la arquitectura de un Sistema de extracción con las características requeridas.

Por otro lado y como un elemento fundamental en esta evaluación, nos apoyaremos en un prototipo funcional, que será construido a estos efectos. El mismo será evaluado en diferentes instancias del proyecto, motivando los resultados obtenidos conclusiones de carácter más general.

1.3 Metodología de trabajo

Para el presente proyecto se eligió un modelo de proceso que se basa en una adaptación del proceso: "The Unified Software Development Process" de Jacobson, Booch y Rumbaugh [JAC], por lo que contará con algunas de sus principales características a saber:

- Ser un proceso iterativo e incremental, permitiendo la mitigación de riesgos en forma temprana y una mejor planificación para todas las fases por las que transcurre el proceso. Estas características son muy apropiadas para el caso de nuestra investigación ya que posibilita la construcción de un prototipo partiendo de funcionalidades básicas, que se incrementarán y mejorarán de manera natural.
- Centrado en la arquitectura, promoviendo la flexibilidad de la misma y el reuso efectivo del software involucrado. En el caso de este proyecto, y como se puede ver la descripción de la arquitectura presentada en la sección 4.2.1, se tendrán módulos bien definidos, con funcionalidades claras, lo que permitirá un desarrollo modular con las ventajas que esto trae consigo.

Las principales actividades involucradas en el proceso fueron:

- Definición y alcance del problema. Descripción de requerimientos. Relevamiento del estado actual de aplicaciones existentes y herramientas utilizadas.
- Descripción de la arquitectura. Diseño del prototipo. Definición de herramientas de software y hardware a utilizar.
- Implementación del prototipo.
- Verificación del prototipo.
- Evaluación de las técnicas empleadas
- Documentación de los resultados obtenidos.

1.3.1 Cronograma de trabajo

De acuerdo a lo dicho anteriormente, el proyecto fue dividido en fases, algunas de las cuales a su vez se subdividieron en diferentes iteraciones. Cada fase y cada iteración tienen sus objetivos definidos, del mismo modo se espera al final de cada fase haber generado ciertos documentos o productos.

Se distinguen tres fases:

- **Fase Inicial (1 mes)**: Se basa en la visión general de los requerimientos del proyecto, rasgos clave y principales restricciones. Incluye la preparación del proyecto, planificación del mismo, análisis del problema y definición de la arquitectura propuesta para el prototipo.

- **Fase Elaboración (6 meses):** Esta etapa, que ocupó la mayor parte del tiempo de duración del proyecto, tuvo como objetivos el diseño y la construcción de un prototipo con la finalidad de servir como herramienta de evaluación de la utilización de las Técnicas de Estado Finito para la generación automática de metadata. Como consecuencia del modelo de proceso elegido y en busca de la obtención de los mayores beneficios, se subdividió la etapa en tres iteraciones en las cuales se enriqueció el comportamiento funcional del prototipo. Asimismo, se efectuaron evaluaciones de los resultados intermedios obtenidos.

- **Fase Evaluación (1 mes):** Consistió fundamentalmente en la evaluación del prototipo final construido, la extracción y documentación de los resultados obtenidos.

En el siguiente diagrama de Gantt se muestra la distribución a lo largo del proyecto de algunas de las principales actividades (Figura Nº 1).

ID	Nombre de la tarea	Comienzo	Finalización	Duración
1	Relevamiento del estado del arte.	02/06/2003	13/06/2003	2w
2	Definición de la arquitectura general.	16/06/2003	30/06/2003	2,2w
3	Elaboración del Plan de Proyecto	16/06/2003	30/06/2003	2,2w
4	Implementación de un prototipo incremental	01/07/2003	30/12/2003	26,2w
5	Evaluación primaria de los resultados	01/09/2003	28/11/2003	13w
6	Testeo del prototipo sobre el dominio elegido	01/12/2003	30/12/2003	4,4w
7	Evaluación de resultados (precision/recall)	01/01/2004	13/01/2004	1,8w
8	Elaboración de informe final	15/10/2003	30/01/2004	15,6w

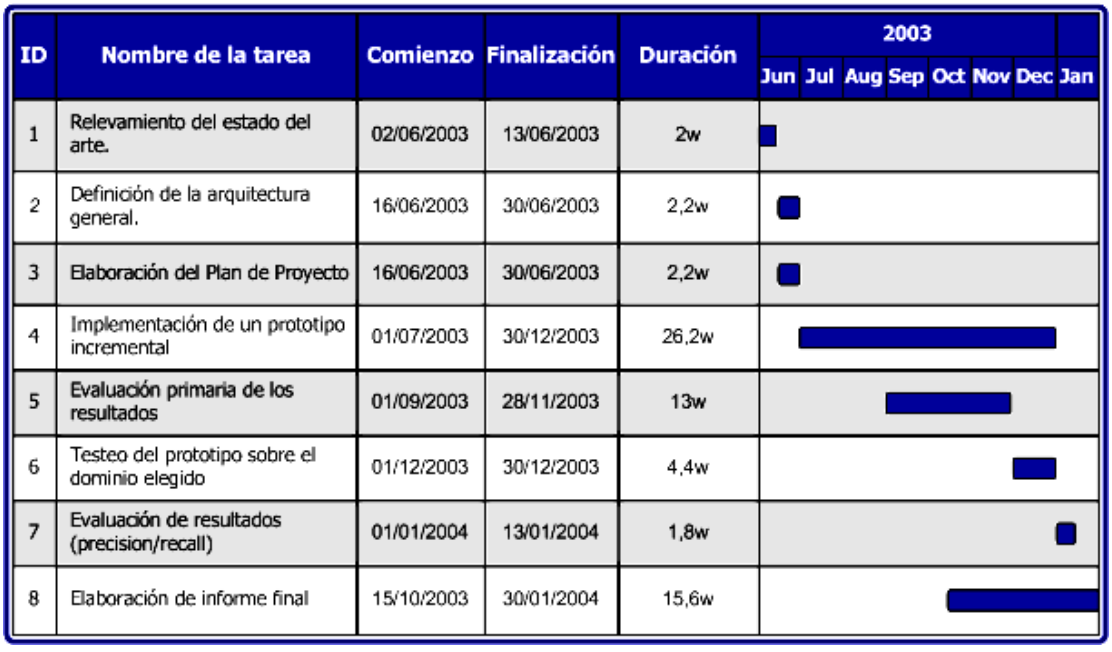


Figura Nº 1 – Distribución de actividades

El Documento de Planificación describe detalladamente el alcance y los resultados esperados de cada una de las fases e iteraciones que componen este proyecto.

1.4 Organización del documento

El informe se encuentra estructurado en 5 capítulos. En los mismos, se pretende ofrecer una visión global al lector de los aspectos más relevantes del proyecto, profundizándose estos temas en los documentos elaborados a lo largo del proyecto. A continuación se resume el contenido de cada uno de los capítulos.

- Capítulo Nº 2.- Se presenta el problema que dio lugar al presente proyecto, introduciéndose los conceptos principales y el contexto en el que se enmarca.

- Capítulo Nº 3.- El capítulo desarrolla los temas relacionados al análisis del problema, mostrando una visión general del estado del arte. Asimismo, se describen a rasgos generales los requerimientos funcionales y no funcionales y se ofrece una visión general de la solución planteada y su alcance.

- Capítulo Nº 4.- Se da una descripción general del diseño del prototipo, incluyendo su arquitectura. Se introducen y motivan las principales decisiones tomadas. Se realiza un estudio de la implementación poniendo énfasis en las tecnologías utilizadas y su interoperabilidad. El capítulo finaliza con la evaluación realizada sobre el prototipo.

- Capítulo Nº 5.- Finalmente, se describen las conclusiones obtenidas, producto del trabajo realizado, y el trabajo futuro.

Al final de este documento se encuentran las Referencias citadas a lo largo del mismo y la Bibliografía consultada.

Capítulo Nº 2: Presentación del problema

2.1 Introducción

El área de Extracción de Información [MUS99] es una sub área del Procesamiento de Lenguaje Natural y una sub área de la Inteligencia Artificial, que se encarga del procesamiento de texto libre arbitrario (como artículos de diarios, revistas, páginas Web, etc.) encontrando elementos específicos de un dominio dado y almacenándolos de una forma más estructurada, como por ejemplo elementos de una base de datos. Dentro de los beneficios que se encuentran al estructurar los textos originales o partes relevantes de los mismos, encontramos la facilidad de manipulación y la posibilidad de generar búsquedas semánticas sobre ellos, que van más allá que las clásicas búsquedas por palabra clave, imponiendo de alguna manera "orden dentro del caos que implica el universo de la información" en la actualidad. Dos factores principales son reconocidos como obstáculos para la expansión de la extracción de la información: la portabilidad implicada en la necesidad de los usuarios de tener sistemas que actúen sobre diversos dominios y escenarios, y la performance. La gran variedad de escenarios posibles requieren la resolución de conjuntos de ambigüedades semánticas y estructurales que tornan de alta complejidad el procesamiento de lenguaje natural.

Con la unión del escenario Web y las herramientas para la extracción de información, encontramos tradicionalmente programas especializados llamados wrappers [KUH02], que se encargan de la identificación de datos relevantes y de mapearlos a un formato adecuado para su posterior utilización. La meta es la utilización de metodologías para desarrollar wrappers genéricos que se puedan adaptar, de una manera (semi) automática, a una clase amplia pero acotada de fuentes de datos relacionadas a diferentes dominios.

Es en este contexto que se plantea el problema a resolver, buscando evaluar las máquinas de estado finito como técnica para el desarrollo de una herramienta que permita de manera eficaz la extracción de datos relevantes, de un documento perteneciente a un dominio particular.

2.2 Conceptos Generales

2.2.1 Introducción a las Técnicas de Estado Finito

Las técnicas de estado finito [BEE99] han sido utilizadas en el correr de estos años en temas vinculados a compilación, procesamiento de voz, reconocimiento de patrones y procesamiento de lenguaje natural entre otros, arrojando resultados alentadores en estas áreas, donde la eficiencia en el procesamiento de las entradas a superado la posible falta de expresividad.

La utilización de los llamados autómatas finitos consolida la base principal para el uso de estas técnicas. Un autómata finito [HOP79] se define como un modelo matemático de un sistema, con entradas y salidas discretas. El sistema puede estar en cualquiera de un número finito de configuraciones internas o estados. El estado del sistema resume la información relativa a las entradas anteriores que es necesaria para determinar el comportamiento del sistema ante subsecuentes entradas. Un autómata finito esta representado por un conjunto finito de estados y un conjunto de transiciones de un estado a otro, las cuales están asociadas a un símbolo de entrada de un alfabeto finito, donde se ha definido una operación de concatenación entre tiras de símbolos.

Los autómatas se comportan como reconocedores de tiras para un lenguaje especificado mediante expresiones regulares, permitiendo determinar si una tira dada pertenece o no al lenguaje.

Los transductores de estado finito se presentan como una ampliación de los autómatas dando la posibilidad de asociar una salida no binaria (tira aceptada o rechazada) al procesamiento. Es decir, dada una tira de entrada y un lenguaje, ofrecen la posibilidad de obtener como salida una tira de símbolos del mismo alfabeto que la entrada u otro según corresponda. Esta tira estará conformada por la concatenación de los símbolos de salida de las transiciones efectivamente realizadas.

Los transductores, utilizados como herramientas de extracción de información, se definen como autómatas finitos con transiciones sobre parejas de símbolos asociados a relaciones regulares (o racionales) y más formalmente se pueden definir como:

Un Transductor de Estado Finito (FST) es una tupla $T=(Q,\Sigma_1, \Sigma_2 ,i,F,E)$, donde:

- Q conjunto finito de estados
- Σ_1, Σ_2 alfabetos de entrada y salida
- i estado inicial, $i \in Q$
- F conjunto finito de estados finales, $F \subseteq Q$
- E conjunto finito de aristas, $E \subseteq Q \times \Sigma_1 \cup \{\varepsilon\} \times \Sigma_2 \cup \{\varepsilon\} \times Q$

Puede definirse una función δ: $Q \times \Sigma_1 \cup \{\varepsilon\} \times \Sigma_2 \cup \{\varepsilon\} \rightarrow 2^Q$, tal que

i) $\delta(q,\varepsilon, \varepsilon) = \{q\}$
ii) $\delta(q,a,b) = \{q' / \exists (q,a,b,q') \in E\}$

Figura Nº 2 – Definición de transductores

El uso del operador reemplazo [KAR94] que permite sustituir una tira del alfabeto por otra bajo cierto contexto, es de especial importancia a la hora de identificar instancias de un lenguaje particular.

Con el fin de evaluar cuantitativa y cualitativamente estas herramientas en el procesamiento de lenguaje natural, se utilizan enfoques asociados al estudio del comportamiento de ciertas medidas sobre un corpus lingüístico determinado. Un corpus es un conjunto de textos almacenados en formato electrónico, y agrupados con el fin de estudiar una lengua o una determinada variedad lingüística.

Tres de las medidas comúnmente utilizadas son: la *precision* (que mide la calidad de los datos reconocidos), el *recall* (que determina la cantidad de datos reconocidos) *y* el *F-score* (que implica una aproximación al significado geométrico ponderado de las medidas anteriores) definidas como:

Precision	=	ocurrencias correctamente identificadas / total de ocurrencias identificadas.
Recall	=	ocurrencias correctamente identificadas / total de ocurrencias.
F – score	=	cociente entre el producto del valor del recall, la precisión y un factor ($ß^2$ +1) y la suma del producto del valor de la precisión y $ß^2$ y el valor del recall.

Figura Nº 3 – Medidas de evaluación de los transductores

Otro factor importante en el estudio de estas herramientas esta dado por la evaluación del tiempo de ejecución que insume el uso de las mismas.

2.2.2 Introducción a las Metadatas

En los últimos años, el incremento sin mesura de las fuentes disponibles en la Web, impuso la necesidad de manejar de forma efectiva la heterogeneidad que abarcan. La dispersión, la dificultad para la localización e integración, el alto factor dinámico que presentan los documentos y la volatilidad, son algunas de las características asociadas a las fuentes de información distribuidas dentro de este ambiente.

Las metadatas constituyen una de las piedras angulares en la búsqueda de la optimización de los procesos de recuperación de información en el escenario antes mencionado, permitiendo la descripción de las fuentes disponibles y soportando una variedad de funciones enfocadas a los mecanismos de búsqueda, como la localización, evaluación, selección y recuperación de datos. Es en esta dirección que se efectúan grandes esfuerzos para lograr mejorar semánticamente la búsqueda de información significativa para un dominio específico, involucrando para este fin distintas áreas.

Existen diferentes formas de especificar la metadata, que alcanzan diferentes grados de expresividad, desde sencillas descripciones hasta estructuras enriquecidas y estrictamente definidas. Esta diversidad ha llevado a la búsqueda de una estandarización para alcanzar una completa optimización a la hora de su utilización.

Comúnmente, se define a la metadata como “datos acerca de los datos” o “información acerca de la información”, definiéndose así una forma de construir conocimiento a partir de los datos de las fuentes de información. Seiner [SEI00] por su parte da una definición más explicita diciendo que “los metadatos son información documentada a través de herramientas de tecnologías de la información que mejoran la comprensión, tanto técnica como comercial, de los datos y de los procesos relacionados con ellos”.

Si nos concentramos en los documentos Web podemos distinguir dos tipos de metadatas: una que es utilizada para ofrecer información para browsers y motores de búsquedas, y otra cuya finalidad es la descripción propia del contenido de la página Web. A su vez, se pueden hallar embebidas en los documentos o se pueden almacenar de forma separada garantizándose su uso a la hora del procesamiento.

La idea de contar con un formato universal para expresar la amplia gama de datos que encontramos en la Web se ha descrito como utópica, debido principalmente a que este formato requeriría un elevado número de descriptores específicos asociados a cada uno de los dominios encontrados. Los esquemas desarrollados con este fin, varían en complejidad, desde listas de elementos descriptores a composiciones de estructuras complejas.

2.2.3 Introducción a las Ontologías

Un paso importante hacia la transformación de la red "desde un espacio de información a un espacio de conocimientos" [NOY01], está dado por el uso de ontologías que permitan la definición de un dominio específico de datos, posibilitándose la generación de las bases para la llamada Web Semántica, con la capacidad de almacenar e inferir conocimiento en lugar de presentar a la Web solo como un contenedor de datos aislados.

Guarino define una ontología como "una estructura semántica intencional que codifica las reglas implícitas restringiendo la estructura de una parte de la realidad", teniendo una ontología formal, estructura lógica para permitir razonamiento sobre los conceptos dentro de la ontología. Gruber por su parte, dio su visión diciendo que "una ontología es una especificación explícita de una conceptualización"

Las ontologías proveen de una manera de representar y compartir el conocimiento utilizando un vocabulario común, permitiendo manejar un formato para intercambiar ese conocimiento, facilitando la reutilización del mismo.

La inserción de las ontologías dentro del ambiente Web ha pasado por varias etapas: inicialmente se buscó un vocabulario común y dar significado a la información que se almacenaba, pero fue cuando se incorporaron los conceptos de jerarquización, especialización y reracionamientos, que se pudo hablar de los comienzos de las ontologías en este medio. Posteriormente las propiedades, restricciones sobre las propiedades, reglas de inferencia y restricciones de clases fueron refinando el concepto, brindando mayor riqueza y posibilitando la creación de lenguajes con mayor expresividad.

Las ontologías se pueden ver como esquemas de metadatos, que proveen un vocabulario controlado de términos, cada uno de ellos definido explícitamente con una semántica procesable por una máquina. Para completar la tarea de las ontologías que definen las teorías de dominios compartidos y comunes, permitiendo acceso en base a contenidos, interoperabilidad y comunicación a través de la Web, se requiere la estandarización de los diversos lenguajes desarrollados para la generación y manipulación de ontologías sobre la Web.

Además de la carencia de una metodología estándar que facilite la creación de ontologías con "buenas propiedades", una de las principales dificultades que se debe de enfrentar, es que un experto dentro del dominio debe ser capaz de desarrollarla y mantenerla manualmente, requiriéndose un gran esfuerzo para que la ontología descrita se mantenga consistente y coherente. Por este motivo, se han realizado grandes esfuerzos enfocados a automatizar o semi automatizar la generación de las mismas, por ejemplo a través de métodos orientados a tener elementos que sirvan como entrenamiento previo para el procesamiento.

2.3 Visión global del problema

Dados un documento y una descripción conceptual del dominio al que pertenece, el problema que se intentará resolver consiste en establecer un relacionamiento entre los conceptos encontrados en la descripción de dominio y las instancias de los mismos dentro del documento ingresado.

Este relacionamiento plantea una serie de inquietudes que deberán ser resueltas para cumplir con el objetivo propuesto:

- Expresividad de la descripción de dominio: estudiar que elementos deberían de estar presentes en la descripción conceptual para establecer las bases necesarias para el relacionamiento.

- Diversidad de conceptos: establecer una clasificación para los conceptos encontrados y proponer los mecanismos necesarios para su manipulación total o parcial.

- Procesamiento del documento: relevar las necesidades de procesamiento del documento ingresado.

- Entrada/salida: determinar el flujo de la información, estableciéndose las particularidades de los documentos que conforman la entrada y la salida del mismo.

Capítulo Nº 3: Análisis

3.1 Introducción

La etapa de análisis embebida en la fase inicial del proyecto, tiene como objetivo principal la visión general de los requerimientos del proyecto, la identificación de los rasgos clave, así como también de las principales restricciones involucradas y la definición del entorno de desarrollo del Proyecto. Inicialmente, dio lugar a la realización de relevamientos relacionados con las tecnologías involucradas en la solución propuesta.

En este capítulo se presentan los aspectos más relevantes involucrados en esta etapa del proceso.

3.2 Visión general del estado del arte

3.2.1 Relevamiento de las metadatas existentes

Las metadatas [IAN97] son consideradas como el principal factor para promover la integración e intercambio de información entre fuentes Web heterogéneas. En un ambiente como Internet deben ser capaces de mejorar los servicios de búsqueda y obtener resultados más precisos.

A partir del estudio del estado del arte efectuado, se observaron una serie de aspectos que vale la pena destacar debido a su importancia a la hora de crear metadatas con las características mencionadas anteriormente.

Un aspecto importante, que no debe perderse de vista en la búsqueda de la estandarización de la generación de metadata descriptiva de los documentos, es la calidad de la misma. Gran parte de los documentos existentes son publicados por usuarios que si bien conocen el domino de la información, no son expertos en el uso de las herramientas que se utilizan para la especificación de metadatas. Este problema, sumado al tiempo requerido para llegar a dominar este tipo de herramientas, hace que muchas veces se generen metadatas de baja calidad que no cumplen los requerimientos que permitirían facilitar la búsqueda y recuperación de la información que describen. Por lo dicho anteriormente, es de interés la automatización del procedimiento de generación de metadata, que permita que usuarios con poco conocimiento de estos lenguajes sean capaces de generar metadatas de alta calidad. Una propuesta para atacar este tipo de problemas es la generación automática de metadata a partir de una descripción del dominio de la información con la que se está trabajando. Para ello se requerirá la participación de un grupo reducido de usuarios expertos que generen está descripción, quiénes sí deberán tener un buen grado de conocimiento de los lenguajes de definición de ontologías. A partir de la utilización de esta descripción, los usuarios inexpertos en la generación de las metadatas podrán enriquecer sus documentos a través de un procedimiento automático y sin necesidad de conocer en profundidad los lenguajes de metadatas, ni la totalidad del dominio en el cual se enmarca el documento.

Por último, y referente al almacenamiento de la metadata, encontramos dos alternativas. La primera presenta la posibilidad de almacenar la metadata embebiéndola en el documento al que hace referencia, mientras que la segunda opción plantea el almacenamiento en forma separada, posiblemente en una base de datos. Cada una de estas opciones brinda ventajas respecto a la otra. La primera permite evitar el problema de mantener un vínculo con los datos que describe, evitando en gran medida las inconsistencias que pudieran generarse al introducir modificaciones a los documentos. La otra opción permite mayor comodidad a la hora de

manipular la metadata y de dar una solución para el caso de objetos que por sus características no puedan contener metadata embebida. De estos dos enfoques, la segunda propuesta es la que parece más atractiva para este prototipo, ya que el mismo está orientado a mejorar la búsqueda a través de la facilidad en la manipulación de la metadata.

RDFS [RDFS, BRO00] se presenta como un lenguaje con gran proyección en Internet y un buen grado de expresividad tanto en la representación de ontologías como en la especificación de la metadata asociada a documentos en la Web. Existen diversos esfuerzos que apuntan a lograr la divulgación y estandarización de este lenguaje, así como también se han desarrollado numerosas propuestas para mapear conceptos expresados en diferentes lenguajes a la sintaxis de RDFS.

Para una visión más completa del relevamiento sobre las metadatas realizado, consultar el Documento de Relevamiento de Metadatas.

3.2.2 Relevamiento de las aplicaciones existentes

La cantidad de información disponible en la World Wide Web se incrementa día a día, ampliando el espectro de datos relevantes para una gran cantidad de dominios de interés para sus usuarios. Muchos investigadores han estudiado procedimientos para extraer datos de documentos semi estructurados y convertirlos a un formato estructurado, que pueda ser fácilmente consultado y en donde se puedan utilizar técnicas del área de base de datos.

La forma más común de extraer datos en la Web se basa en la generación de wrappers, ya sean manuales, semi automáticos, o totalmente automáticos. La mayoría de estas propuestas en mayor o menor medida usan delimitadores o tags HTML que limitan el texto a ser extraído haciendo a las herramientas sensibles ante cambios en el formato de la página examinada.

Existe un vasto número de propuestas para la generación de wrappers, que pueden agruparse, según los autores Laender y Ribeiro-Neto [LAE02], de acuerdo a la técnica principal que usen para su generación, destacándose: aquellas que desarrollan lenguajes para los wrappers, las que utilizan las marcas provistas por el formato HTML, las basadas en el procesamiento de lenguaje natural, las que utilizan algoritmos de inducción, las que recurren a un modelo para la extracción, y aquellas que se ayudan mediante el uso de ontologías o modelos conceptuales.

Fuertemente vinculado al enfoque que se utilice para la generación de los wrappers está el tipo de los documentos a los cuales se orientará la aplicación. Encontramos en estos, diferentes grados de estructuración, que van desde texto libre a otros altamente estructurados. Existe una relación natural entre la metodología que se utilice para la generación de los wrappers y las características de los documentos. Asimismo, cabe destacar la estrecha vinculación que se presenta entre el tipo de documento utilizado como fuente de información y el grado de automatización que logran alcanzar las aplicaciones. Documentos pobres en estructura son generalmente procesados por herramientas con una fuerte componente de pre procesamiento, mediante el cual usuarios expertos en el dominio de acción, determinan, examinan y/o etiquetan un conjunto de documentos para la conformación de los denominados corpus de entrenamiento.

Considerando un enfoque centrado en el uso de ontologías para la descripción de un dominio de información dado, se muestra que las herramientas son capaces de lograr un alto grado de automatización en la generación de los wrappers, pero es requerida la participación de un usuario que defina la ontología. Existe una fuerte relación entre la expresividad de la ontología y el grado de automatización que es posible alcanzar. Este tipo de aplicación resulta conveniente a la hora de analizar documentos que si bien son del mismo dominio, presentan diferencias en cuanto a su formato. Es uno de los enfoques que se adapta mejor a los cambios inherentes al medio en el que se encuentren los documentos.

El análisis efectuado, determinó la existencia de tres ideas fundamentales que marcan una tendencia hacia la optimización de los sistemas de extracción de información.

Por un lado observamos la importancia de realizar un pre-procesamiento sintáctico y semántico cuando se trabaja con documentos que carecen de un alto grado de estructuración, en el que se identifiquen conceptos de tipos que puedan ser relevantes para el resto del proceso de extracción de información.

Por otra parte la idea de manejar mediante árboles la estructura del documento, facilita la manipulación de la información contenida en el mismo mejorando los niveles de performance del sistema.

Por último el uso de ontologías para describir el dominio de información relevante permite lograr buenas cualidades en cuanto a generalidad del dominio y adaptación a cambios.

Para una visión más completa del relevamiento sobre las herramientas de extracción de información realizado, consultar el Documento de Relevamiento de Aplicaciones Existentes.

3.3 Requerimientos

Para cumplir el objetivo propuesto, se hizo hincapié en el desarrollo de un prototipo que consolidara la utilización de máquinas de estado finito y el uso de modelos conceptuales en una aplicación orientada a la extracción automática de metadata asociada a un documento particular.

En los siguientes dos puntos se resume el problema visto en términos de requerimientos funcionales y no funcionales.

3.3.1 Requerimientos funcionales

- *Procesamiento de un modelo conceptual para un dominio dado.*

Procesamiento de la descripción asociada a un dominio y obtención, a partir de ese análisis, de los elementos necesarios para la generación de la metadata asociada a un documento perteneciente al dominio.

- *Generación de metadata para un documento.*

A partir de los elementos obtenidos en el proceso descrito en el punto anterior, debe permitirse el procesamiento de un documento de html escrito en idioma español, y la generación de metadata descriptiva del mismo.

- *Almacenamiento de metadata.*

El Sistema deberá brindar al usuario final la posibilidad de almacenar dicha metadata embebida en el documento de origen, o generar otro documento conteniéndola, el cual deberá estar expresado en algún lenguaje de especificación válido.

- *Interfaz gráfica*.

Se deberá construir una interfaz gráfica que permita la cómoda explotación de las funcionalidades provistas por el Sistema para satisfacer los puntos anteriores.

3.2.2 Requerimientos no funcionales

Técnica a utilizar

Se tiene como requerimiento la utilización de las Técnicas de Estado Finito como herramienta fundamental para llevar adelante el proceso de generación de la metadata.

Modularidad

El Sistema a ser construido deberá presentar un alto grado de modularidad que permita la independencia funcional de sus componentes.

Documentación

Se entregará un manual de usuario en forma escrita, que explique todas las funcionalidades y la forma de realizar las operaciones, el cuál sea claro, detallado y cubra todas las funcionalidades del prototipo.

Automatización

El Sistema permitirá resolver los problemas en forma automática, y la interacción con el usuario estará limitada a acciones del tipo de seleccionar una descripción de dominio o un documento de entrada.

Tiempo de respuesta

No se tienen requerimientos puntuales en cuanto a tiempo de respuesta, sin embargo se valorará la eficiencia en el proceso de generación de metadatas, a la vista de que las técnicas a utilizar para dicho fin producen en algunos casos tiempo de ejecución importantes.

Por información más detallada sobre el los requerimientos funcionales y no funcionales, ver el Documento de Especificación de Requerimientos.

3.4 Descripción general de la solución

La solución propuesta para la generación automática de la metadata, que analizaremos en un alto nivel de abstracción en esta sección, toma como entradas un documento definiendo la descripción asociada al dominio a ser utilizado y una página en formato HTML perteneciente al dominio ingresado, como muestra la Figura Nº 4.

Como salida del flujo de información obtendremos la metadata, enriqueciendo de esta manera, el documento ingresado, mediante la identificación de los conceptos relevantes mencionados en la descripción del dominio.

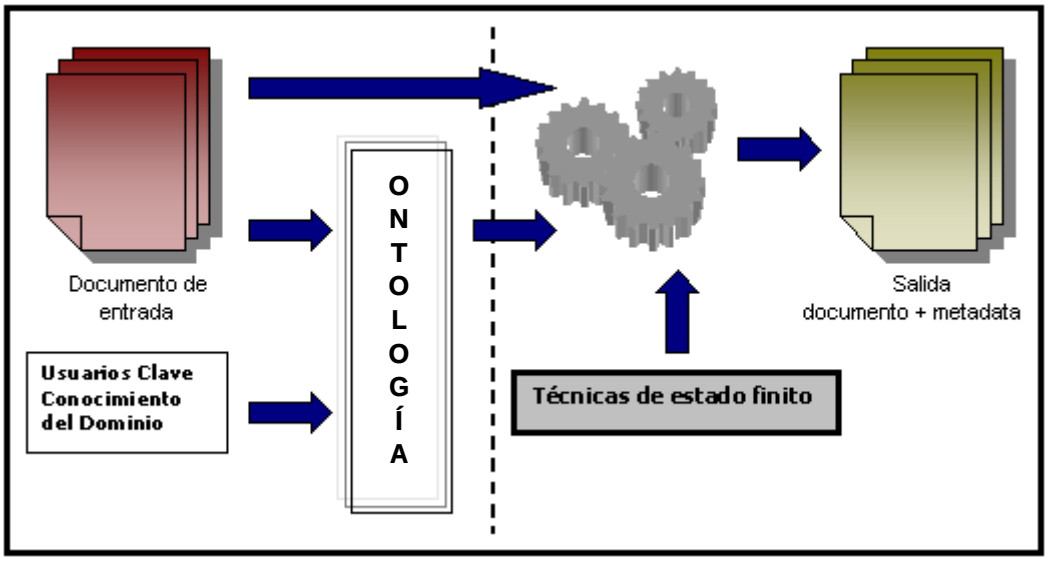


Figura Nº 4 – Entradas y salidas de la solución propuesta

Una vez realizados los relevamientos correspondientes y el análisis de los requerimientos impuestos para el desarrollo del prototipo, se optó por el uso de ontologías para la descripción del dominio estudiado. Se utilizó RDFS como lenguaje para esta descripción del dominio y de la metadata asociada al documento ingresado. Esta elección se basó principalmente en la amplia difusión, facilidades y propiedades que brinda RDFS como descriptor. RDFS, ampliación de RDF [RDF], se obtiene agregando al lenguaje original, primitivas adicionales que dotan a RDFS de una mayor expresividad.

En cuanto a las características del motor para el prototipo, se buscó un diseño que siguiera con las buenas prácticas detectadas en el relevamientos efectuados. Las tres ideas fundamentales en la que se basan estas prácticas son:

- Pre procesamiento sintáctico y semántico cuando se trabaja con documentos que carecen de un alto grado de estructuración, en el que se identifiquen conceptos de tipos que puedan ser relevantes para el resto del proceso de extracción de información.
- Utilización de una estructura interna de árbol para representar la descripción del dominio, facilitando la manipulación de la información contenida en el mismo.
- Utilización de ontologías para describir el dominio de información relevante, factor que repercute en la generalidad del dominio y la adaptación a cambios.

Asimismo, como mencionamos en la sección anterior, el requerimiento principal de este motor es que se base principalmente en la utilización de técnicas de estado finito para lograr su cometido.

Por lo dicho anteriormente, se dividirá el funcionamiento del motor en 4 etapas bien diferenciadas. Estas etapas mostradas en la Figura Nº 5, se describen a continuación.

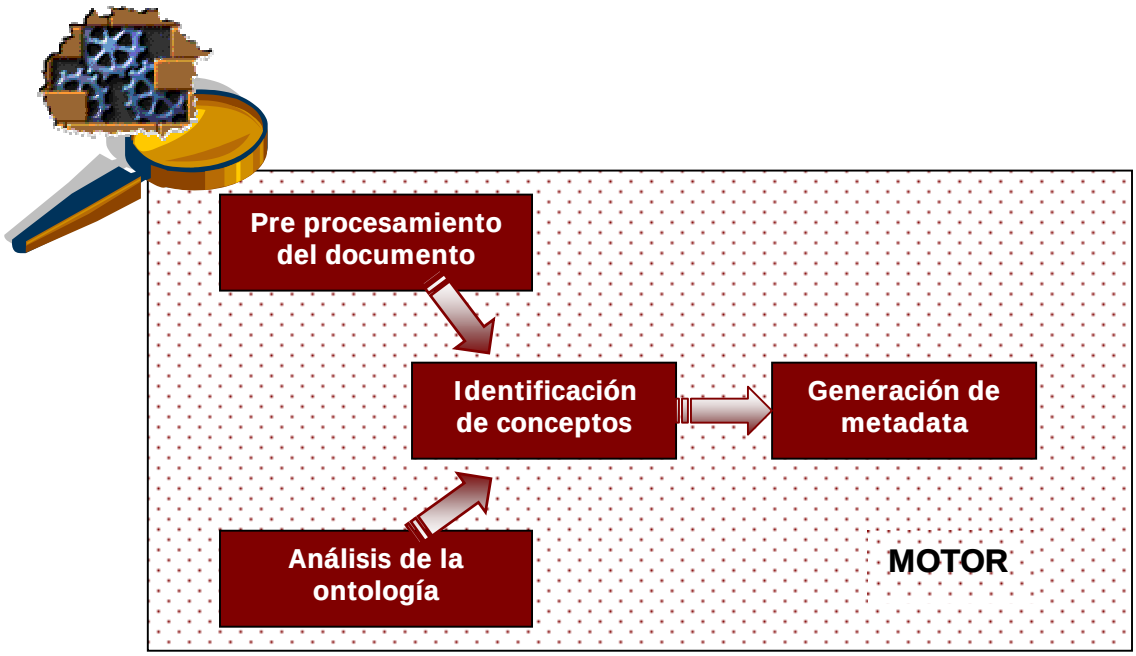


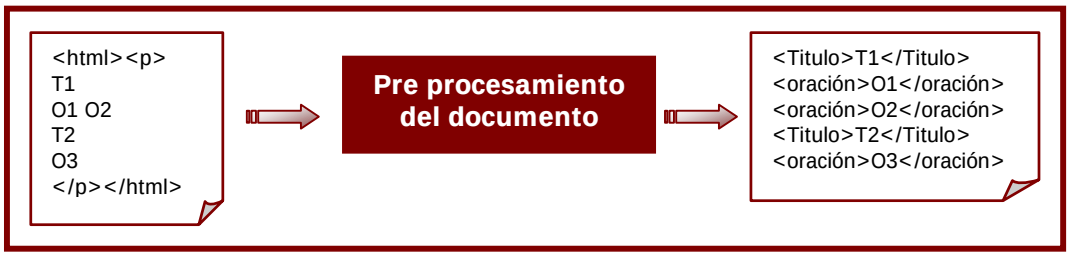
Figura Nº 5 – Visión general de la solución propuesta

- Análisis de la ontología ingresada

En esta etapa, el motor se encargará de estudiar la ontología ingresada, extrayendo de la misma los elementos necesarios para el correcto funcionamiento del resto del proceso. Asimismo, se encargará de darle una estructura interna a la información recolectada.

- Pre procesamiento del documento dado

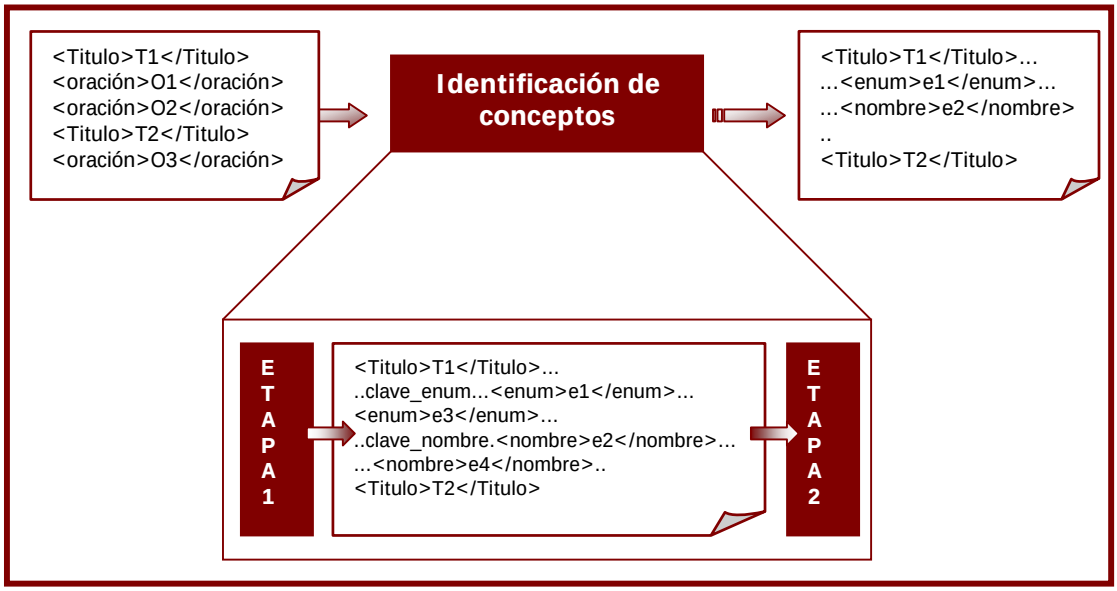
El análisis de la estructura de la página ingresada se efectúa en esta etapa del procesamiento. La idea es buscar aquellos elementos estructurales de las páginas ingresadas que puedan complementar y apoyar las decisiones tomadas por etapas siguientes a la hora de detectar los elementos de la metadata. Además, como parte de esta etapa, el procesamiento determinará y eliminará las porciones de la página que no aporten elementos dentro del procesamiento, tal es el caso de la información referente a la presentación de la página.



- Identificación de conceptos dentro del documento

Esta etapa se dividió en dos grandes tareas. La primera esta involucrada con la búsqueda en la página, ahora estructurada, de posibles instancias válidas de los dominios asociados a los conceptos de la ontología. Estos dominios se dividen en dos grandes grupos, teniendo por un lado aquellos que son detectados automáticamente por el Sistema (unidades, números, fechas, horas, nombres, lugares, organizaciones, direcciones Web, direcciones de e-mail), y por otro lado los dominios que están dados como una enumeración de sus instancias constituyentes, localizados en una estructura auxiliar a la que el Sistema deberá de acceder (archivo de texto, planilla, base de datos, etc.).

La segunda tarea involucra la vinculación de los elementos identificados anteriormente con los conceptos de la metadata, en base a ciertos elementos que darán el contexto que deberá de analizarse. Dichos elementos estarán presentes en la descripción del dominio, como un listado de palabras claves, el cual se podrá enriquecer a través del uso de herramientas que permitan la obtención de sinónimos o conjugaciones verbales a partir de los elementos listados



- Generación de la metadata

La etapa final del proceso, se encargará de darle formato a la información reconocida dentro de la página ingresada.

3.4.1 Ejemplo

A continuación se presentará un ejemplo concreto que pretende clarificar las ideas previamente manejadas.

Sea un dominio simplificado vinculado con la cartelera teatral de Montevideo. En él, se describen obras de teatro las cuales tienen un nombre, un genero y una duración asociada. Asimismo, las obras están vinculadas a personas en el rol de directores o actores.

La siguiente figura representa la especificación en RDFS asociada al dominio propuesto.

```
<rdf:RDF>
        <rdf:Description ID="Obra">
                <rdf: type resource = "http://www.w3.org/2000/01/rdf-schema#class">
        </rdf:Description>
        <rdf:Description ID="Persona">
                <rdf: type resource = "http://www.w3.org/2000/01/rdf-schema#class">
        </rdf:Description>
        <rdfs:Property rdf:ID="titulo">
                <rdfs: comment>
                        titulo, nombre, titulada, denominada, llamada, nombrada
                </rdfs: comment>
                <rdfs: domain rdf:resource= "#Obra">
                <rdfs: range rdf:resource = "titulo_obra">
        </rdfs:Property>
        <rdfs:Property rdf:ID="genero">
                <rdfs: comment>
                        género, tipo, clase
                </rdfs: comment>
                <rdfs: domain rdf:resource= "#Obra">
                <rdfs: range rdf:resource = "genero">
        </rdfs:Property>
        <rdfs:Property rdf:ID="duracion">
                <rdfs: comment>
                        Duración, largo, tiempo, minutos
                </rdfs: comment>
                <rdfs: domain rdf:resource= "#Obra">
                <rdfs: range rdf:resource="numero">
        </rdfs:Property>
        <rdfs:Property rdf:ID="dirige">
                <rdf: type rdf:resource= "seq">
                <rdfs: comment> director, dirige, dirección </rdfs: comment>
                <rdfs: domain rdf:resource= "#Obra">
                <rdfs: range rdf:resource = "nombre">
        </rdfs:Property>
        <rdfs:Property rdf:ID="actua">
                <rdf: type rdf:resource= "bag">
                <rdfs: comment>
                        Interpretes, actor, actúa, actriz, protagonizada, elenco
                </rdfs: comment>
                <rdfs: domain rdf:resource= "#Obra">
                <rdfs: range rdf:resource = "nombre">
        </rdfs:Property>
</rdf:RDF>
```

Dada esta especificación, la etapa de análisis de la ontología debería de extraer la siguiente información:

- los datos relativos a la jerarquía de los conceptos presentados
- los datos asociados al dominio (etiqueta domain) y rango (etiqueta range) de cada una de las propiedades involucradas
- las palabras claves para reconocer posibles contextos asociados a instancias de los conceptos, embebidas dentro de la etiqueta comment.

Por otra parte, un ejemplo de documento asociado a este dominio (extraído de http://cartelera.uruguaytotal.com/) , será la entrada a la etapa de pre procesamiento en la cual se indicaran diferentes elementos que ayudarán a la realización del procesamiento principal.

Barro Negro

Género: Tragi-comedia / Duración: 90 minutos / Calificación: +12 años
Autor: Gabriel Núñez
Dirección: Marcelino Duffau
La obra se desarrolla a bordo de un ómnibus de recorrido urbano, donde el espectador-pasajero asiste a diversas historias que tienen lugar en el trayecto. La marcha se interrumpe en un boliche especialmente adaptado para el espectáculo. Mezcla de actuación, bebidas y baile en torno a una historia de amor, lo convierten en un espectáculo distinto, sin precedentes, que ahora ingresa en su temporada número 13.

Las novias de Travolta

Género: Comedia
Autor: Andrés Tulipano
Dirección: Álvaro Ahunchain
Elenco: Roxana Blanco, Ana Pañella, Carmen Morán y María de la Paz
Cuatro amigas se reúnen para el cumpleaños de Gaby, la primera de ellas en llegar a los 40. Gaby es divorciada, y tiene un hijo adolescente. Cris cumplió el sueño de sus padres: es médica, y ha dedicado toda su vida a la profesión. Estela se casó muy joven, y recién ahora está descubriendo cómo funciona el mundo real. Lucía se había marchado a Suecia con sus padres, durante la década del ´70; su regreso a Montevideo es para ella como un viaje al pasado.

La etapa de identificación de instancias, utilizará los datos asociados a los dominios descritos en la ontología, para identificar las instancias válidas localizadas en el documento ingresado. En el ejemplo y como muestra la siguiente figura, esta etapa intentará localizar instancias asociadas a los tipos titulo_obra, genero, numero y nombre, siendo los dos primeros dominios de tipo enumerado en los cuales se deberá ingresar información adicional que enumere sus posibles instancias, mientras que los dos últimos son dominios básicos que el Sistema puede resolver mediante la aplicación de reglas.

<titulo_obra>Barro Negro</titulo_obra>

Género: **<genero>**Tragi-comedia**</genero>** / Duración: **<numero>**90**</numero>** minutos / Calificación: +12 años
Autor: **<nombre>**Gabriel Núñez**</nombre>**
Dirección: **<nombre>**Marcelino Duffau**</nombre>**
La obra se desarrolla a bordo de **<numero>**un**<numero>** ómnibus de recorrido urbano, donde el espectador-pasajero asiste a diversas historias que tienen lugar en el trayecto. La marcha se interrumpe en **<numero>**un**<numero>** boliche especialmente adaptado para el espectáculo. Mezcla de actuación, bebidas y baile en torno a **<numero>**una**<numero>** historia de amor, lo convierten en **<numero>**un**<numero>** espectáculo distinto, sin precedentes, que ahora ingresa en su temporada número **<numero>**13**<numero>**.

<titulo_obra >Las novias de Travolta</titulo_obra >

Género: **<genero>**Comedia**</genero>**
Autor: **<nombre>**Andrés Tulipano**</nombre>**
Dirección: **<nombre>**Álvaro Ahunchain**</nombre>**
Elenco: **<nombre>**Roxana Blanco**</nombre>**, **<nombre>**Ana Pañella**</nombre>**, **<nombre>**Carmen Morán**</nombre>** y **<nombre>**María de la Paz**</nombre>**
<numero>Cuatro**</numero>** amigas se reúnen para el cumpleaños de **<nombre>**Gaby**</nombre>**, la primera de ellas en llegar a los **<numero>**40**</numero>**. **<nombre>**Gaby**</nombre>** es divorciada, y tiene **<numero>**un**</numero>** hijo adolescente. **<nombre>**Cris**</nombre>** cumplió el sueño de sus padres: es médica, y ha dedicado toda su vida a la profesión. **<nombre>**Estela**</nombre>** se casó muy joven, y recién ahora está descubriendo cómo funciona el mundo real. **<nombre>**Lucía**</nombre>** se había marchado a Suecia con sus padres, durante la década del ´70; su regreso a Montevideo es para ella como **<numero>**un**</numero>** viaje al pasado.

Asimismo, esta etapa culminará con la utilización de la información extraída de los posibles contextos para reconocer cuales de las instancias identificadas pertenecen efectivamente a la metadata asociada al documento.

<titulo>Barro Negro</titulo>
Género: **<genero>**Tragi-comedia**</genero>** / Duración: **<duración>**90**<duración>** minutos / Calificación: +12 años
Autor: Gabriel Núñez
Dirección: **<dirige>**Marcelino Duffau**</dirige>**
La obra se desarrolla a bordo de un ómnibus de recorrido urbano, donde el espectador-pasajero asiste a diversas historias que tienen lugar en el trayecto. La marcha se interrumpe en un boliche especialmente adaptado para el espectáculo. Mezcla de actuación, bebidas y baile en torno a una historia de amor, lo convierten en un espectáculo distinto, sin precedentes, que ahora ingresa en su temporada número 13.

<titulo>Las novias de Travolta</titulo>

Género: **<genero>**Comedia**</genero>**
Autor: Andrés Tulipano
Dirección: **<dirige>**Álvaro Ahunchain**</dirige>**
Elenco: **<actúa>**Roxana Blanco**</actúa>**, **<actúa>**Ana Pañella**</actúa>**, **<actúa>**Carmen Morán**</actúa>** y **<actúa>**María de la Paz**</actúa>**
Cuatro amigas se reúnen para el cumpleaños de Gaby, la primera de ellas en llegar a los 40. Gaby es divorciada, y tiene un hijo adolescente. Cris cumplió el sueño de sus padres: es médica, y ha dedicado toda su vida a la profesión. Estela se casó muy joven, y recién ahora está descubriendo cómo funciona el

Finalmente la etapa de generación de la metadata, consolidará los elementos identificados conformándose las instancias asociadas al documento ingresado. La siguiente figura muestra la metadata asociada al ejemplo.

```
<?xml version = "1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"   xmlns:ont="..\Ontologia.txt#">
  <rdf:Description rdf:about="..\Documento.txt">

  <ont:Obra rdf:ID="id0">
    <ont:titulo>Barro Negro</ont:titulo>
    <ont:duracion>90</ont:duracion>
    <ont:genero>Tragi-comedia</ont:genero>
    <ont:dirige>
        <rdf:seq>
          <rdf:li>Marcelino Duffau</rdf:li>
        </rdf:seq>
    </ont:dirige>
  </ont:Obra>

  <ont:Obra rdf:ID="id1">
    <ont:titulo>Las novias de Travolta</ont:titulo>
    <ont:genero>Comedia</ont:genero>
    <ont:dirige>
        <rdf:seq>
          <rdf:li> Álvaro Ahunchain </rdf:li>
        </rdf:seq>
    </ont:dirige>
    <ont:actua>
        <rdf:bag>
          <rdf:li> Roxana Blanco</rdf:li>
          <rdf:li> Ana Pañella</rdf:li>
          <rdf:li> Carmen  Morán</rdf:li>
          <rdf:li> María de la Paz</rdf:li>
        </rdf:bag>
    </ont:actua>
  </ont:Obra>
  </rdf:Description>
</rdf:RDF>
```

3.5 Alcance del prototipo

El alcance del prototipo a construir abarca las funcionalidades descriptas en la sección anterior, incluyendo además la generación de una interfaz gráfica para el usuario final.

Dentro de la funcionalidad de identificación de conceptos dentro del documento, solo se incluirá la posibilidad de acceder a definiciones de tipos enumerados a través de un archivo de texto. Además, no se considerará el enriquecimiento de los listados de palabras clave a través del análisis de sinónimos o conjugaciones verbales.

Capítulo Nº 4: Prototipo

4.1 Introducción

Como se señaló en los capítulos anteriores, uno de los objetivos de este proyecto es la construcción de un prototipo, el cual se utilizará como herramienta fundamental para la evaluación de las Técnicas de Estado Finito como mecanismo para la extracción de metadata.

En este contexto, en el Capítulo Nº 3 se puntualizaron los requerimientos funcionales y no funcionales que se definen para dicho prototipo. En dicho capítulo se presentan además, los lineamientos principales de la solución propuesta para el problema que se ataca, es decir la generación automática de la metadata asociada a un documento. Sobre dichos lineamientos generales es que se construye este prototipo.

En este capítulo se busca resumir los principales aspectos referentes al diseño, implementación y evaluación del prototipo construido. Se dará una idea del funcionamiento de cada uno de los componentes, a la vez que se explicará en los casos que corresponda, como estos componentes utilizan las herramientas de estado finito.

4.2 Diseño

En esta sección nos concentraremos en el diseño del prototipo, analizando el mismo en diferentes niveles de abstracción, abarcando la definición de la arquitectura propuesta, la división en subsistemas, la organización en paquetes y el diseño de las clases.

4.2.1 Arquitectura

En este punto mostraremos una visión general de la arquitectura que se definió para el prototipo. Como veremos, en esta visión se identifican los principales componentes del Sistema, las entradas y salidas del mismo, y en general los flujos de información entre los componentes reconocidos.

La arquitectura definida toma como base un conjunto de propiedades consideradas clave para un buen diseño que se detallan a continuación:

- Abstracción: Estudio del problema planteado desde varios niveles de abstracción.
- Modularidad: Estudio de la arquitectura buscando una definición modular para el Sistema a ser construido.
- Búsqueda de las cualidades principales de un buen diseño: organización en capas, realizando la división entre los aspectos de presentación, lógica de la aplicación y persistencia, visibilidad, robustez y escalabilidad, uso de interfaces bien definidas y uso de mecanismos estándar.
- Independencia funcional: Aplicar los conceptos de modularidad, abstracción y ocultamiento de información para desarrollar módulos con una función claramente definida y una interacción mínima con los otros módulos constituyentes.
- Extensibilidad: Se busca un diseño que permita la incorporación sin mayor esfuerzo de nuevas funcionalidades.

La Figura Nº 6 muestra una vista general de la arquitectura. La zona marcada en gris delimita el motor lógico de la aplicación, englobando aquellos componentes que serán los encargados de implementar la funcionalidad básica del prototipo. Se puede apreciar que dicho motor recibe como entradas una ontología que define un domino particular, y un documento en formato HTML perteneciente al dominio de la ontología. La salida del motor está compuesta por un documento conteniendo la metadata asociada al documento de entrada. Asimismo, se identifican algunos componentes auxiliares que sirven de apoyo para la principal funcionalidad del prototipo.

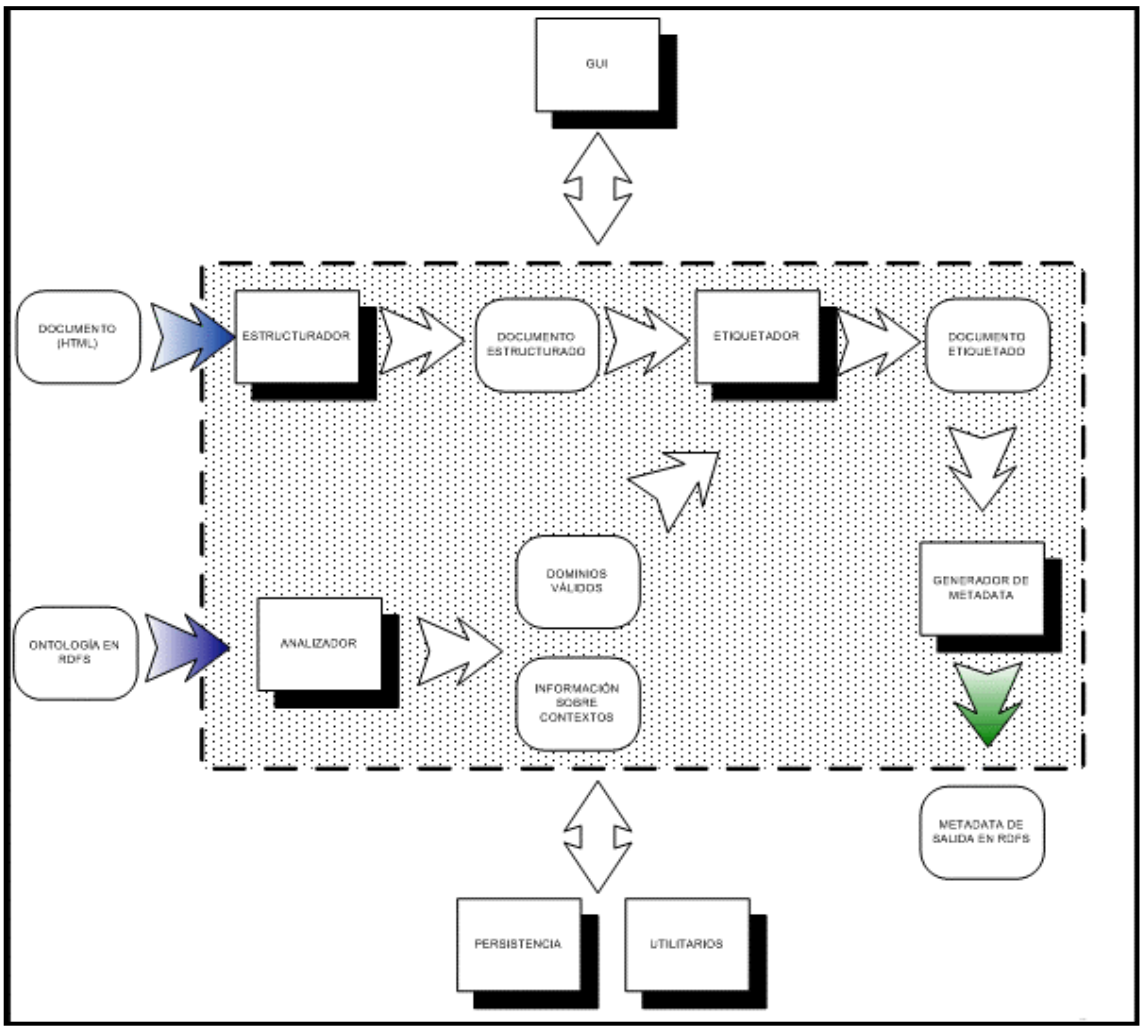


Figura Nº 6 – Vista de la arquitectura

A continuación brindamos una breve descripción de la funcionalidad asociada a cada uno de los componentes en los que se divide la arquitectura propuesta, y de los flujos de información establecidos entre los mismos.

- **Analizador** (Figura Nº 7): El objetivo de este componente es el análisis de la ontología ingresada con el fin de extraer de la misma aquellos elementos necesarios para el resto del proceso de generación de la metadata. Estos elementos consisten principalmente en el conjunto de dominios válidos presentes en la ontología y el conjunto de los elementos claves para el análisis de contextos de instancias de los dominios anteriores identificadas en el documento de entrada.

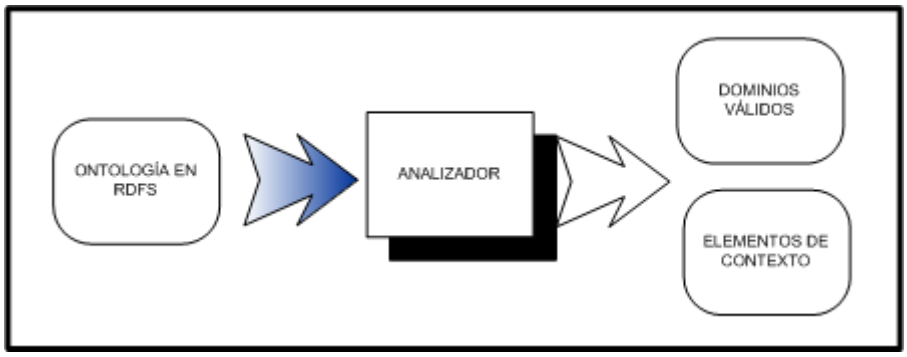


Figura Nº 7 - Analizador

- **Estructurador** (Figura Nº 8): este componente tiene como objetivo principal la preparación, mediante la incorporación de marcas especialmente definidas, del documento de entrada (en formato HTML), detectando la información correspondiente a títulos, párrafos u oraciones, categorías gramaticales y estructuras embebidas. La generación de un documento estructurado permitirá alcanzar una entrada más "segura" para el Etiquetador, en el sentido en que nos permitirá de forma temprana detectar elementos (como lo son los títulos y las tablas) que involucran una forma diferente de detección de elementos de dominio.

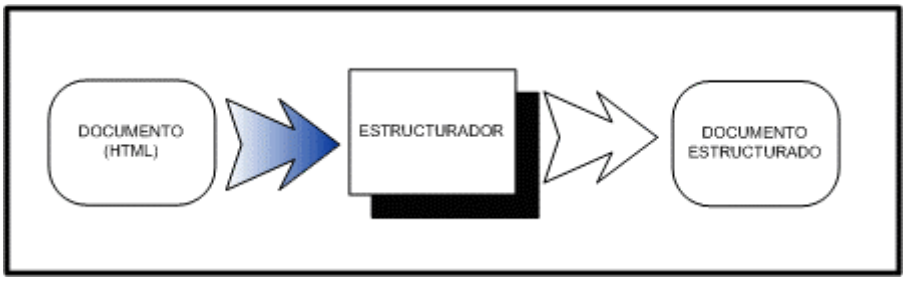


Figura Nº 8 - Estructurador

- **Etiquetador** (Figura Nº 9): Este componente realiza la tarea de indicar dentro del documento estructurado, las instancias de elementos de la ontología. Para esta tarea utiliza la información generada por el Analizador referida tanto a dominios válidos dentro de la ontología, como a palabras clave para el análisis de contexto de las instancias encontradas por él. Los dominios que manejará la aplicación se dividen en dos grandes categorías: aquellos que el Sistema detecta automáticamente mediante la aplicación de un conjunto de reglas asociadas a ese dominio, y aquellos que están dados como reglas basadas en una enumeración de sus elementos constituyentes. Dentro de la primera categoría encontraremos los siguientes dominios básicos: números, nombres, fechas, horas, lugares, unidades, organizaciones, direcciones web y direcciones de e-mail. Los elementos de la segunda categoría son detectados o bien por formar parte de la ontología ingresada o por encontrarse almacenados en alguna estructura auxiliar accedida por la aplicación (base de datos, un archivo de texto, o una planilla de cálculo).

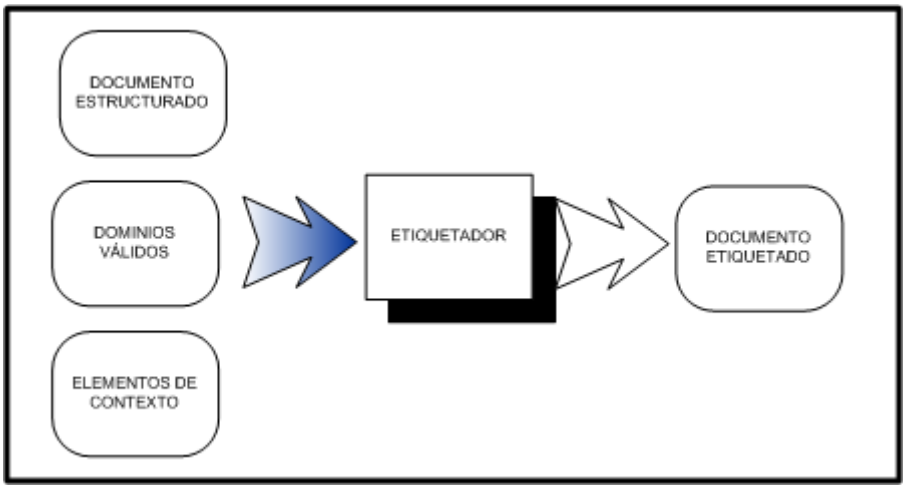


Figura Nº 9 - Etiquetador

- **Generador** (Figura Nº 10): Este componente, involucrado en la etapa final del proceso, realiza la tarea de aplicar al documento etiquetado, un conjunto de reglas para establecer la vinculación de los elementos detectados y consolidar de esta manera la metadata asociada al documento.

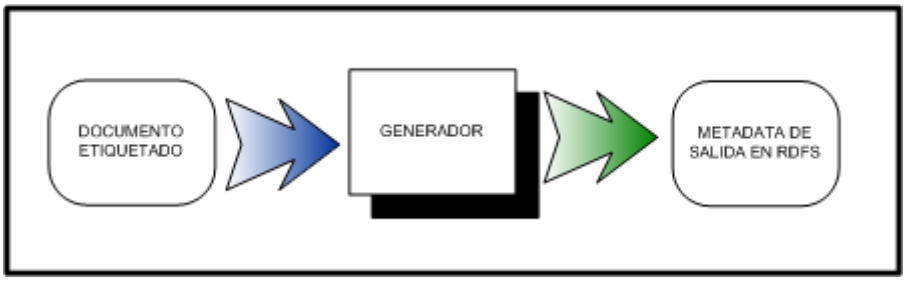


Figura Nº 10 - Generador

- **GUI**: Implementa la interfaz gráfica con el usuario.
- **Persistencia:** Actúa como repositorio estático de información y es utilizado por los componentes principales del Sistema.
- **Utilitarios**: Brinda diversos servicios auxiliares utilizados por los componentes principales del Sistema.

El documento de Descripción de la Arquitectura presenta una visión más detallada, en la que se incluye el alcance de la misma para cada una de las iteraciones en las que se dividió la construcción de este prototipo.

4.2.2 Descomposición en Subsistemas:

Los diferentes componentes incluidos en la descripción de la arquitectura pueden agruparse de acuerdo a la funcionalidad que cumplen, en varios subsistemas. A continuación se muestra esta clasificación en subsistemas, indicando las dependencias establecidas entre ellos (Figura Nº 11).

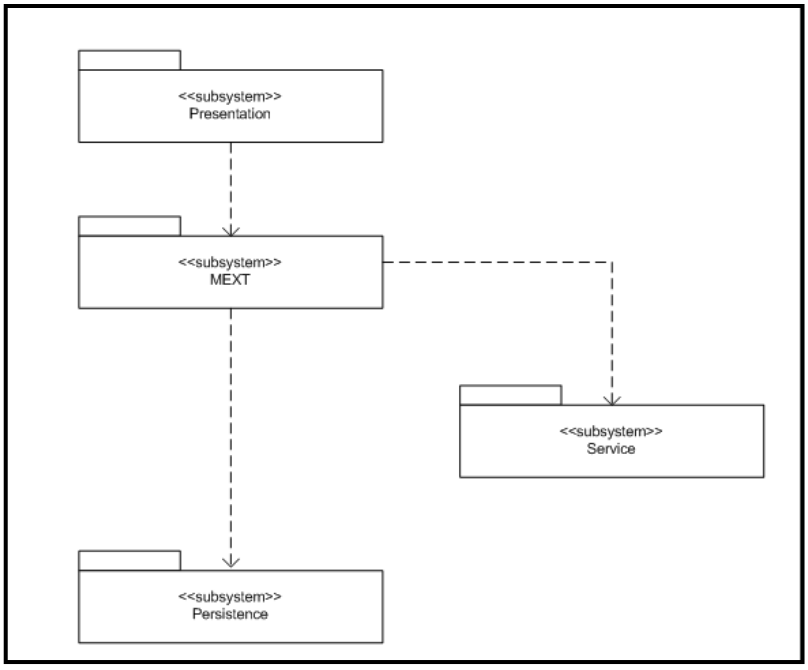


Figura Nº 11 – Descomposición en subsistemas

A continuación se presentan brevemente cada uno de los subsistemas detectados, analizando el alcance de los mismos.

Subsistema: Presentation

Este subsistema está integrado por los componentes encargados de obtener los datos brindados por el usuario (localización de las páginas, ubicación de la ontología asociada al dominio y localización de los elementos necesarios para la definición de las instancias validas de algún tipo particular) y de realizar la presentación gráfica de los resultados.

Subsistema: MEXT

Subsistema conformado por los componentes que soportan la funcionalidad principal del sistema. Representa el motor lógico del prototipo.

Subsistema: Service

Subsistema compuesto por módulos auxiliares que brindan servicios de bajo nivel principalmente al subsistema MEXT, separando los mismos de la lógica de la aplicación.

Subsistema: Persistence

Subsistema vinculado al almacenamiento estático de ciertos elementos requeridos por los demás componentes para efectuar el procesamiento.

4.2.3 Diagrama de paquetes

La división en subsistemas presentada en el punto anterior se corresponde de manera natural con una vista en capas de los componentes de la arquitectura. Cada uno de los subsistemas corresponde a una de las capas que se muestran en la Figura Nº 12. Aquí se presenta una visión esquemática de los paquetes que componen cada subsistema y la interacción existente entre los mismos.

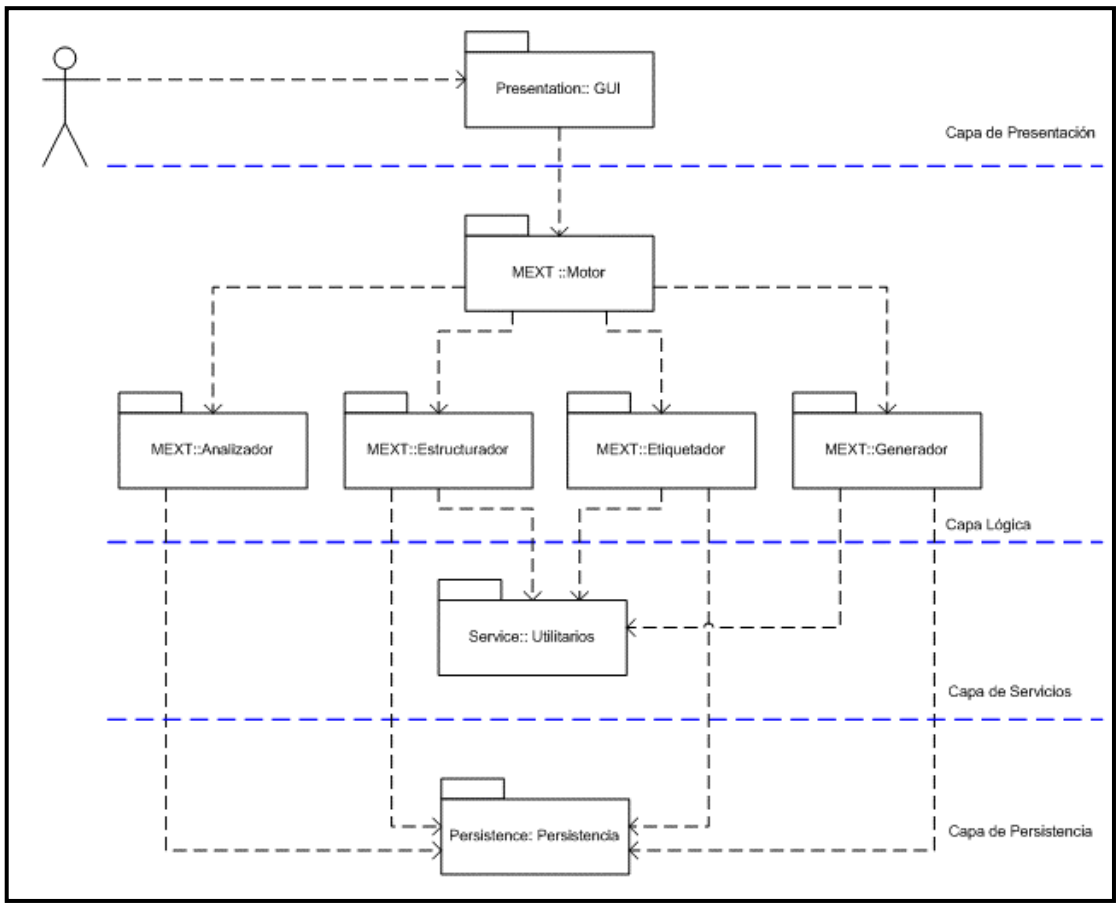


Figura Nº 12 – Diagrama de paquetes

4.2.4 Diagrama de clases

Hemos visto el diseño del Sistema en varios niveles de abstracción, desde una vista general de la arquitectura, hasta una división en paquetes. En esta sección completamos el análisis del diseño, bajando hasta el nivel de las principales clases del Sistema.

A continuación se presenta un diagrama de clases correspondiente al sistema en su totalidad (Figura Nº 13). En él se pueden identificar las clases pertenecientes a los diversos paquetes incluyendo sus principales funciones y atributos.

Por razones de legibilidad y claridad del diagrama se omiten funciones constructoras, selectoras y seteadoras de atributos, así como algunos atributos auxiliares de las clases. También se omiten las funciones y atributos de las clases correspondientes al paquete GUI.

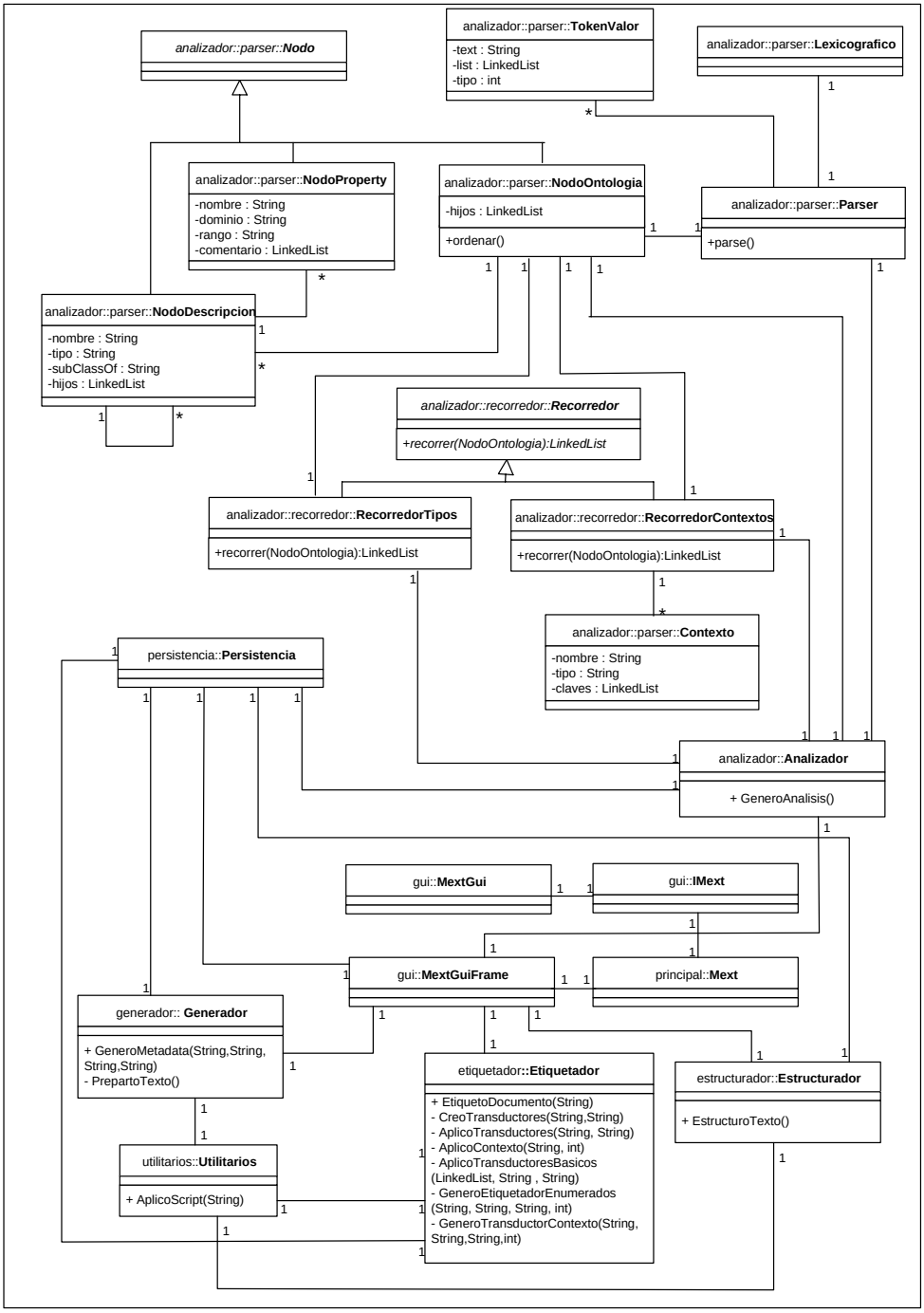


Figura Nº 13 – Diagrama de clases

4.3 Implementación

En esta sección veremos las principales características de la implementación y el funcionamiento de cada uno de los componentes del Sistema. Analizaremos además las herramientas y tecnologías utilizadas, discutiendo los motivos que llevaron a la selección de las mismas para la implementación del prototipo, mostrando como fueron utilizadas en los diferentes componentes.

4.3.1 Herramientas y tecnologías utilizadas

La creación y manipulación de transductores, base de la implementación de varios de los módulos de este prototipo, requiere de la utilización de herramientas de software específicas. En el caso de este proyecto se utilizaron dos herramientas, las cuales se encuentran disponibles en el Instituto de Computación de la Facultad de Ingeniería. Estas herramientas se describen brevemente a continuación:

- XeLDA [XEL] – Motor lingüístico que provee servicios para transformar, normalizar y extraer información de un texto, construido como parte de las investigaciones en el área de lenguaje natural del Centro Europeo de Investigación de XEROX.

- XFST [XFS] – Herramienta para crear y manipular máquinas de estado finito (autómatas y transductores). Desarrollada por el Centro Europeo de Investigación de XEROX.

Dichas herramientas existen en versiones tanto para plataforma Unix como para plataforma Windows, sin embargo en el caso del presente proyecto solamente se disponía de las mismas para ser utilizadas en uno de los servidores Unix de la Facultad.

Dada esta restricción en cuanto a la disponibilidad de las herramientas mencionadas, se optó por Unix como la plataforma sobre la cual correrá el prototipo. El software construido puede compilarse y ejecutarse con funcionalidades limitadas en un ambiente Linux o Windows. En estos casos es posible trabajar con la interfaz gráfica y realizar el análisis y almacenamiento de ontologías, no siendo posible la generación de la metadata, ya que este proceso requiere la utilización de herramientas disponibles para plataforma Unix.

Por otra parte, JAVA [JAV] fue escogido como lenguaje de programación para este prototipo ya que:

- Es un lenguaje ampliamente difundido, multiplataforma, lo que permite la generación de un software portable y fácilmente integrable o extensible.

- Brinda la flexibilidad necesaria para la construcción de este prototipo.

- Presenta buenas características para la generación de la interfaz gráfica.

Si bien parte del desarrollo fue realizado sobre plataforma Linux, el ambiente de desarrollo mayoritariamente utilizado fue JBuilder 7 [JBU], sobre plataforma Windows.

Como herramientas adicionales para el análisis de la descripción de dominio ingresada, se utilizaron:

- JLex [JLE] – Herramienta para la generación automática de analizadores lexicográficos. A partir de una especificación basada en expresiones regulares, realiza la generación de código JAVA correspondiente al analizador.

- CUP [CUP] – Herramienta para la generación automática de parsers. A partir de una especificación basada en una gramática, realiza la generación del código JAVA correspondiente a la misma.

La utilización de estas herramientas surge de forma natural, ya que las mismas están asociadas a la generación de código JAVA, lenguaje que como dijimos fue el seleccionado para la implementación del prototipo.

En la sección 4.3.2 describiremos cada uno de los componentes que integran el Sistema, explicando como estos utilizan las diferentes herramientas mencionadas anteriormente.

4.3.1.1 Sobre la ejecución

Como mencionábamos en el punto anterior, el software corre sobre plataforma Unix. Para la ejecución del prototipo se utilizó el servidor "Lulu" del Instituto de Computación de la Facultad de Ingeniería, que cuenta con las herramientas de software necesarias. Este servidor, con sistema operativo Solaris 7, modelo Sun Enterprise 250, cuenta con un procesador UltraSPARC II de 350 MHz y 512 Mb de memoria RAM.

Ante esta restricción surge el interés de poder utilizar el software en forma remota, tanto desde equipos de plataforma Linux como Windows. La solución propuesta para esto, implica la utilización de las siguientes herramientas (Figura Nº 14):

- SSH [SSH] - Herramienta que permite conexión remota a un equipo Unix, siendo cliente para Linux o Windows..

- Exceed [EXC] o X-Win32 [XWI] – Herramientas que permiten la visualización de aplicaciones gráficas que corren en equipos Unix, desde una máquina con Windows (Servidores X).

A través de una conexión SSH es posible desde una máquina con Linux, conectarse al servidor Lulu, y ejecutar la aplicación con su funcionalidad completa.

En el caso de los equipos con plataforma Windows y de forma similar puede ejecutarse la aplicación a través de una conexión SSH. Si bien es posible correr la misma desde un shell del sistema, no es posible correr la interfaz gráfica. Para solucionar este punto y previo al establecimiento de la conexión, es necesario levantar en el equipo con Windows uno de los servidores X mencionados. Esto permite crear un túnel encriptado por el cual se envíe la información gráfica que será interpretada y desplegada por el servidor X de la máquina remota.

En los casos en los que se corre la aplicación en su modalidad gráfica desde un equipo remoto a través de SSH, se tienen tiempos de respuesta (en el despliegue de las pantallas, no en la generación de la metadata la cual aún en estos casos corre en Lulu) variables en función de la velocidad de la conexión establecida. En el caso de estar trabajando desde una red local, si bien se producen demoras en el tiempo de respuesta de la GUI, las mismas permiten la utilización de la aplicación de forma razonable. Por el contrario, en el caso de conexiones a través de líneas mal lentas, por ejemplo a través de un módem, los tiempos de respuesta sufren demoras importantes. Esto es una consecuencia de que el envió de las pantallas se realiza de forma

encriptada a través de la conexión SSH, lo cual requiere del encriptado y desencriptado de cada pantalla enviada a través del túnel establecido.

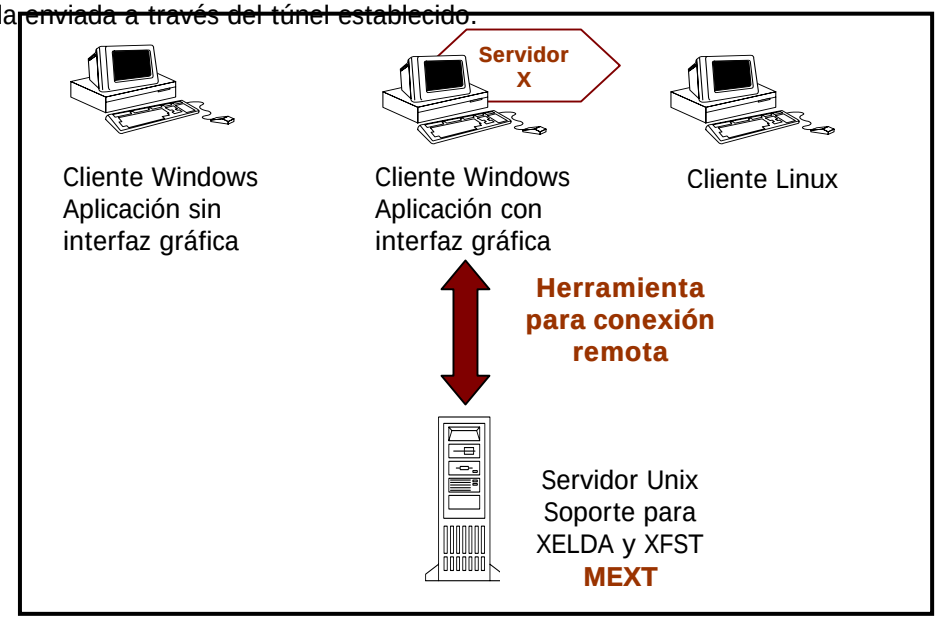


Figura Nº 14 - Ejecución en forma remota

4.3.1.2 Sobre los transductores

Como mencionamos anteriormente, la implementación de algunos de los módulos del Sistema (Estructurador y Etiquetador), esta basada en máquinas de estado finito, más concretamente en transductores. Los mismos son creados a partir de una especificación utilizando la herramienta XFST. Un transductor generado de esta forma puede ser aplicado directamente a un archivo de entrada, generando un archivo de salida resultado de las reglas impuestas en la especificación del mismo sobre la entrada. La Figura Nº 15 esquematiza el funcionamiento descrito, mostrando como ejemplo un transductor sencillo.

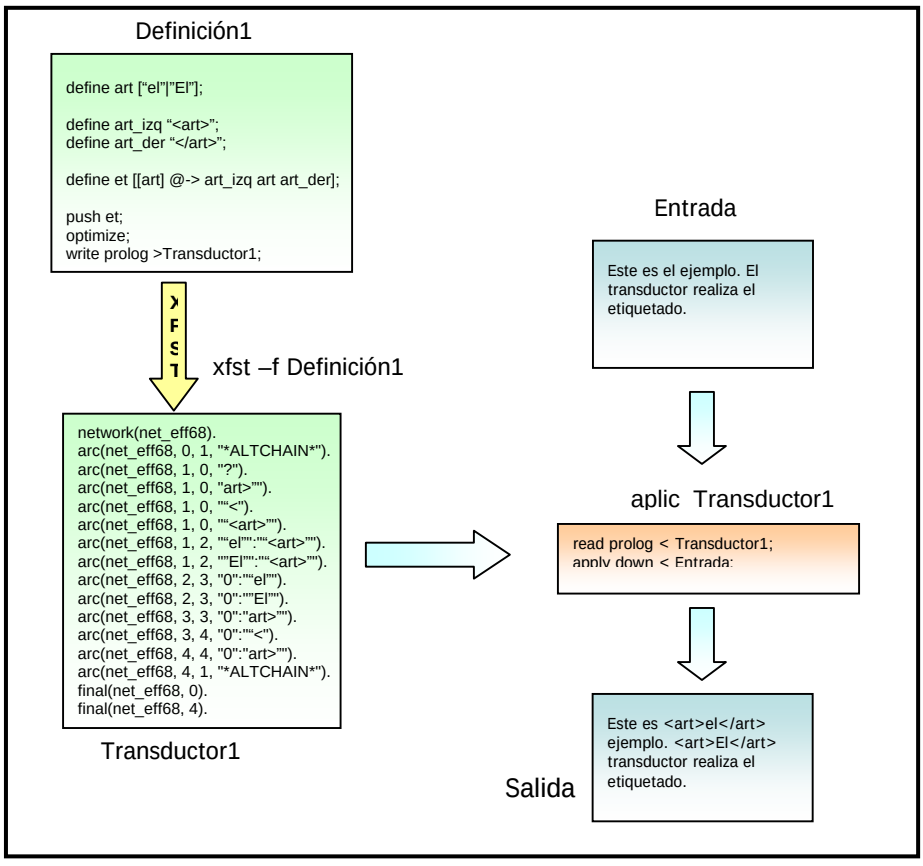


Figura Nº 15 - Ejemplo de aplicación de un transductor

Los módulos de este prototipo que utilizan transductores, basan su funcionamiento en la aplicación en cascadas de los mismos, como se muestra en la Figura Nº 16. A partir de un archivo de entrada, se aplican sucesivamente los transductores, creando archivos intermedios, hasta generar en última instancia el archivo de salida. La aplicación sucesiva de dichos transductores es efectuada por un script, el cual es ejecutado mediante servicios de bajo nivel provistos por el componente Utilitarios.

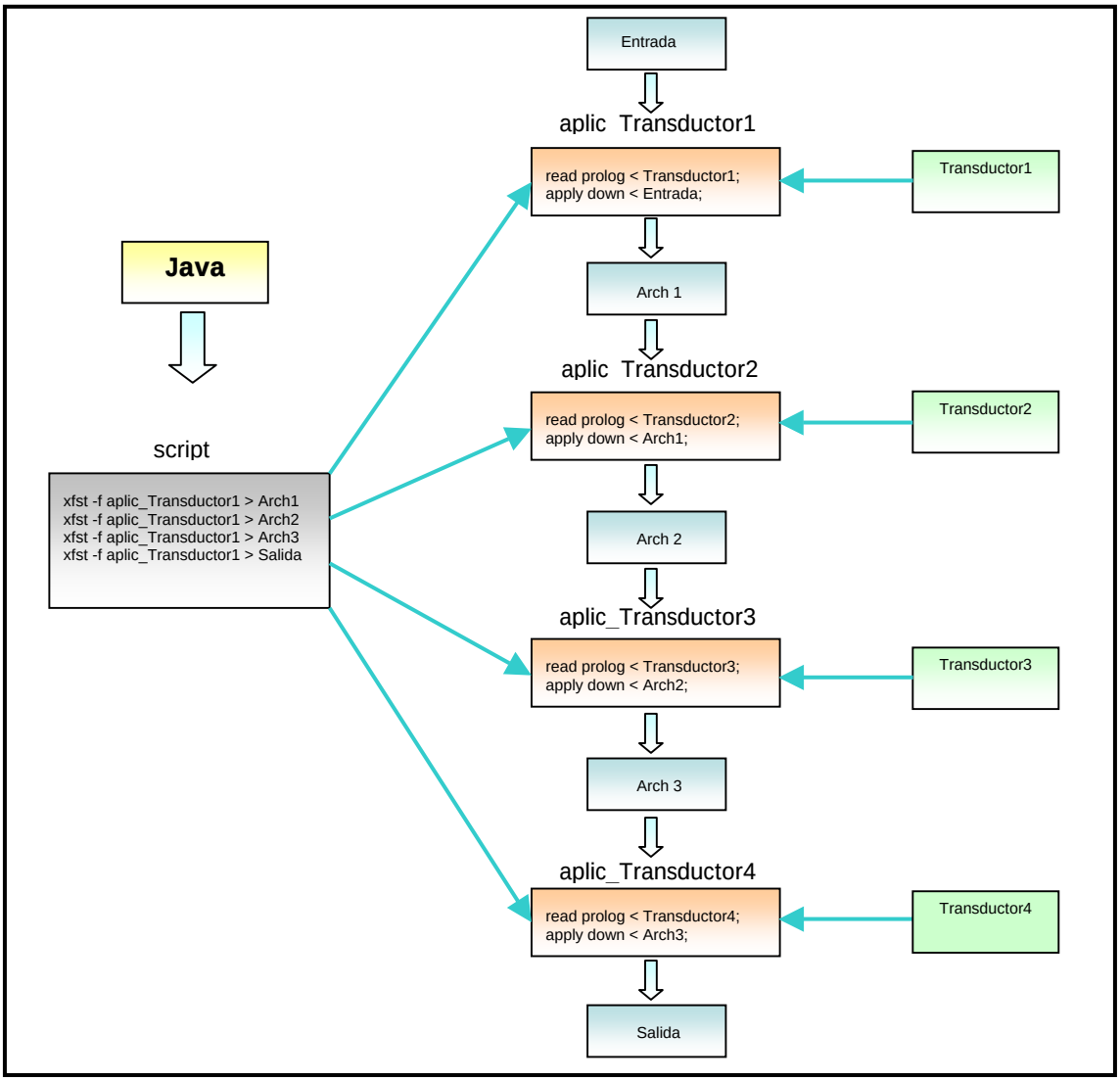


Figura Nº 16 - Ejemplo de aplicación de cascadas de transductores

Cabe destacar que existen en el prototipo construido dos tipos de transductores:

- Por un lado existen transductores tales que el etiquetado que realizan no depende del documento que se está procesando, o de las entradas del Sistema. Es decir, que el transductor no cambia para diferentes entradas. Este tipo de transductores son especificados y generados utilizando la herramienta XFST estáticamente (al igual que los scripts y archivos de aplicación de los mismos) y no en tiempo de ejecución. Ejemplos de estos transductores son los que utiliza el Estructurador para el etiquetado de oraciones, tarea que no depende de cada documento de entrada particular.

- Por otra parte existen transductores cuya definición cambia en cada ejecución de la aplicación, dependiendo de características propias del documento u ontología considerados. Son ejemplo de este tipo los transductores pertenecientes al Etiquetador que realizan el reconocimiento de los tipos enumerados. Estos transductores no pueden ser generados previo a la ejecución, ya que es imposible disponer de la información necesaria para su construcción en el momento de la especificación de los mismos. En estos casos, los archivos de definición (así como los scripts y archivos de aplicación de los transductores) son generados en tiempo de ejecución utilizándose procedimientos en Java para la generación de estos archivos que son compilados utilizando XFST y ejecutados mediante scripts, igual que en el caso anterior.

4.3.2 Descripción de los módulos

En secciones anteriores hemos presentado los diferentes componentes del Sistema, y descrito la funcionalidad de cada uno de ellos, así como sus interacciones.

Nos proponemos ahora a brindar una visión de la implementación de dichos componentes, analizando el funcionamiento de los mismos y discutiendo en algunos casos las decisiones tomadas en torno a ellos.

4.3.2.1 Analizador

Este módulo está encargado de analizar las ontologías ingresadas, obteniendo de ellas los diferentes elementos y la información necesaria para dotar al Etiquetador y al Generador de la capacidad de generar la metadata asociada a documentos pertenecientes a los dominios descritos en las ontologías.

El Analizador recibe como entrada una ontología descriptiva del dominio que se desee trabajar, y obtendrá como salida una lista de tipos válidos asociados a los conceptos descritos en la ontología y una lista de nodos que contendrán la información necesaria para efectuar el análisis de contextos.

La funcionalidad está claramente definida en dos aspectos principales, el parseo de la ontología y la generación y manipulación de las estructuras que contienen los elementos a ser utilizados por otros módulos. De acuerdo a esta división funcional, el Analizador se separa de dos sub módulos que implementan cada una de estas capacidades.

Parser

Este componente es el encargado de realizar el parseo de la ontología que define el dominio. A partir de este análisis se crea una estructura de árbol que oficia de representación intermedia y contiene la información de la ontología relevante a los efectos de la creación de la metadata, en un formato más apropiado para que el Recorredor pueda obtener de ella los elementos necesarios para generar la salida del Analizador.

Los elementos de la ontología que el Parser deberá almacenar en la estructura de árbol son los siguientes:

- Conceptos o clases identificados en la ontología con su respectiva organización jerárquica.

- Relaciones o propiedades vinculadas a los conceptos identificados anteriormente, donde cada relación tiene asociada una lista de palabras clave que nos servirán para identificar las ocurrencias de la misma.

La Figura Nº 17 pretende ayudarnos a identificar con que elementos de la sintaxis de RDFS asociaremos a cada uno de los elementos mencionados en los dos puntos anteriores. En dicha figura se muestra un ejemplo de ontología completa expresada en RDFS donde se observan dos tipos de definiciones:

- Definiciones embebidas entre las etiquetas <rdf:Description> y </rdf:Description> o <rdfs:Class> y </rdfs:Class> correspondientes a conceptos del dominio definidos en la ontología (por ejemplo los señalados en amarillo en la figura). Respecto a este tipo de definiciones podemos decir que las etiquetas “Description” y “Class” nos ayudan a identificar la presencia de un objeto del dominio en la ontología, mientras que la etiqueta clave “subClassOf” es de utilidad para identificar la jerarquía entre los conceptos del dominio.

- Definiciones embebidas entre las etiquetas <rdfs:Property> y </rdfs:Property>, que corresponden a relaciones entre conceptos del dominio que define la ontología, o que también pueden verse como propiedades de los objetos (por ejemplo las señaladas en verde en la figura). La etiqueta “Property” nos ayuda a identificar este tipo de elementos, la etiqueta “Domain” nos indica el concepto al que está asociada la propiedad, la etiqueta “Range” determina el tipo de valores que podrá tomar la propiedad mientras que la etiqueta “Type” ayuda a determinar el tipo de contenedor para la propiedad. Por otro lado, existe la etiqueta “Comment” que nos permite almacenar una lista de palabras claves que serán utilizadas por el Etiquetador para efectuar un análisis del contexto de las instancias de los dominios válidos identificadas previamente.

```
<rdf:RDF>
        <rdf:Description ID="Pelicula">
                <rdf: type resource : "http://www.w3.org/2000/01/rdf-schema#class">
        </rdf:Description>
        <rdf:Description ID="Persona">
                <rdf: type resource : "http://www.w3.org/2000/01/rdf-schema#class">
        </rdf:Description>
        <rdf:Description ID="En_Espaniol">
                <rdf: type resource : "http://www.w3.org/2000/01/rdf-schema#class">
                <rdf: subClassOf resource: "#Pelicula">
        </rdf:Description>
        <rdf:Description ID="No_En_Espaniol">
                 <rdf: type resource : "http://www.w3.org/2000/01/rdf-schema#class">
                 <rdf: subClassOf resource: "#Pelicula">
        </rdf:Description>
        <rdf:Description ID="Number">
                <rdf:type resource:"http://www.w3.org/2000/01/rdf-schema#Class">
        </rdf:Description>
        <rdf:Description ID="Integer">
                 <rdf:type resource:"http://www.w3.org/2000/01/rdf-schema#Class">
                 <rdfs:subClassOf rdf:resource:"#Number">
        </rdf:Description>
        <rdfs:Property rdf:ID="titulo">
                <rdfs: comment>
                        titulo, nombre, titulada, denominada, llamada, nombrada
                </rdfs: comment>
                <rdfs: domain rdf:resource: "#Pelicula">
                <rdfs: range rdf:resource : "http://www.fing.edu.uy/pgtrweb/classes#Titulo">
        </rdfs:Property>
        <rdfs:Property rdf:ID="genero">
                <rdfs: comment>
                        genero, tipo, clase
                </rdfs: comment>
                <rdfs: domain rdf:resource: "#Pelicula">
                <rdfs: range rdf:resource : "http://www.fing.edu.uy/pgtrweb/classes#Genero">
        </rdfs:Property>
        <rdfs:Property rdf:ID="pais">
                <rdfs: comment>
                        pais, nacionalidad, procedencia, origen
                </rdfs: comment>
                <rdfs: domain rdf:resource: "#Pelicula">
                <rdfs: range rdf:resource : "http://www.fing.edu.uy/pgtrweb/classes#Pais">
        </rdfs:Property>
        <rdfs:Property rdf:ID="duracion">
                <rdfs: comment>
                        duracion, largo, tiempo, minutos
                </rdfs: comment>
                <rdfs: domain rdf:resource: "#Pelicula">
                <rdfs: range rdf:resource : "http://www.w3.org/2000/03/example/classes#Integer">
        </rdfs:Property>
        <rdfs:Property rdf:ID="idioma">
                <rdfs: comment>
                        idioma, lenguaje, hablada
                </rdfs: comment>
                <rdfs: domain rdf:resource: "#No_En_Espaniol">
             <rdfs: range rdf:resource : "http://www.fing.edu.uy/pgtrweb/classes#Idioma">
        </rdfs:Property>
        <rdfs:Property rdf:ID="tituloorigen">
                <rdfs: comment>
                        titulo, nombre, origen, procedencia
                </rdfs: comment>
                <rdfs: domain rdf:resource: "#No_En_Espaniol">
                <rdfs: range rdf:resource : "http://www.fing.edu.uy/pgtrweb/classes#TituloOrigen">
        </rdfs:Property>
</rdf:RDF>
```

Figura Nº 17 – Ejemplo de ontología expresada en RDFS

De acuerdo a la sintaxis de RDFS y a los elementos que nos interesa identificar, como acabamos de explicar, se define la gramática que será utilizada por el Parser para procesar la ontología. Dicha gramática aparece simplificada en la Figura Nº 18.

```
/* gramatica */

start with ontologia_rdfs;

ontologia_rdfs ::= ABRE RDF DOS_PUNTOS RDF CIERRA lista_declaraciones ABRE BARRA RDF DOS_PUNTOS RDF CIERRA
                 | ABRE RDF DOS_PUNTOS RDF CIERRA ABRE BARRA RDF DOS_PUNTOS RDF CIERRA

lista_declaraciones ::= ABRE declaracion
                      | lista_declaraciones ABRE declaración

declaracion ::=  RDFS DOS_PUNTOS PROPERTY RDF DOS_PUNTOS ID IGUAL COMILLAS ALFANUMERICO COMILLAS CIERRA property
      | RDFS DOS_PUNTOS PROPERTY RDF DOS_PUNTOS ID IGUAL COMILLAS ALFANUMERICO COMILLAS BARRA CIERRA
      | RDF DOS_PUNTOS DESCRIPTION ID IGUAL COMILLAS ALFANUMERICO COMILLAS CIERRA description
      | RDF DOS_PUNTOS DESCRIPTION ID IGUAL COMILLAS ALFANUMERICO COMILLAS BARRA CIERRA
      | RDFS DOS_PUNTOS CLASS ID IGUAL COMILLAS ALFANUMERICO COMILLAS CIERRA class
      | RDFS DOS_PUNTOS CLASS ID IGUAL COMILLAS ALFANUMERICO COMILLAS BARRA CIERRA
      | RDF DOS_PUNTOS ALFANUMERICO ID IGUAL COMILLAS ALFANUMERICO COMILLAS CIERRA ALFANUMERICO ABRE BARRA RDF
        DOS_PUNTOS ALFANUMERICO CIERRA
      | RDF DOS_PUNTOS ALFANUMERICO ID IGUAL COMILLAS ALFANUMERICO COMILLAS ALFANUMERICO BARRA CIERRA
      | RDFS DOS_PUNTOS ALFANUMERICO ID IGUAL COMILLAS ALFANUMERICO COMILLAS CIERRA ALFANUMERICO ABRE BARRA
        RDFS DOS_PUNTOS ALFANUMERICO CIERRA
      | RDFS DOS_PUNTOS ALFANUMERICO ID IGUAL COMILLAS ALFANUMERICO COMILLAS ALFANUMERICO BARRA CIERRA

/* Descripciones*/

description ::=  ABRE BARRA RDF DOS_PUNTOS DESCRIPTION CIERRA
               | cuerpo_description ABRE BARRA RDF DOS_PUNTOS DESCRIPTION CIERRA
class ::=  ABRE BARRA RDFS DOS_PUNTOS CLASS CIERRA
          | cuerpo_description ABRE BARRA RDFS DOS_PUNTOS CLASS CIERRA
cuerpo_description ::= inicio_description type_subclassof
                       | cuerpo_description inicio_description type_subclassof
                       | inicio_description etiqueta
                       | cuerpo_description inicio_description etiqueta
inicio_description  ::= ABRE RDF DOS_PUNTOS
                       | ABRE RDFS DOS_PUNTOS
type_subclassof ::=  type
                    | subclassof
etiqueta ::= ALFANUMERICO RESOURCE IGUAL COMILLAS direccion COMILLAS CIERRA
           | ALFANUMERICO CIERRA ABRE BARRA ALFANUMERICO CIERRA
type ::=  TYPE RESOURCE IGUAL COMILLAS direccion COMILLAS CIERRA
         | TYPE CIERRA direccion ABRE BARRA TYPE CIERRA
subclassof ::=  SUBCLASSOF RESOURCE IGUAL COMILLAS direccion COMILLAS CIERRA
              | SUBCLASSOF RDF DOS_PUNTOS RESOURCE IGUAL COMILLAS direccion COMILLAS CIERRA
              | SUBCLASSOF CIERRA direccion ABRE BARRA SUBCLASSOF CIERRA


/* Propiedades */

property ::=  ABRE BARRA RDFS DOS_PUNTOS PROPERTY CIERRA
            | cuerpo_property ABRE BARRA RDFS DOS_PUNTOS PROPERTY CIERRA
cuerpo_property ::= inicio_property comment_range_domain
                    | inicio_property2 type_prop
                    | cuerpo_property inicio_property comment_range_domain
                    | cuerpo_property inicio_property2 type_prop
                    | inicio_property etiqueta
                    | cuerpo_property inicio_property etiqueta
inicio_property ::= ABRE RDFS DOS_PUNTOS
inicio_property2 ::= ABRE RDF DOS_PUNTOS
comment_range_domain ::=  comment  | range   | domain
comment ::=  COMMENT CIERRA cuerpo_comment ABRE BARRA RDFS DOS_PUNTOS COMMENT CIERRA
           | COMMENT CIERRA ABRE BARRA RDFS DOS_PUNTOS COMMENT CIERRA
cuerpo_comment ::= termino_comment
                  | termino_comment COMA cuerpo_comment
termino_comment::= ALFANUMERICO
domain ::=  DOMAIN RDF DOS_PUNTOS RESOURCE IGUAL COMILLAS direccion COMILLAS CIERRA
range ::=  RANGE RDF DOS_PUNTOS RESOURCE IGUAL COMILLAS direccion COMILLAS CIERRA
type_prop ::= TYPE RDF DOS_PUNTOS RESOURCE IGUAL COMILLAS direccion COMILLAS CIERRA
direccion ::= termino_direccion
             | termino_direccion direccion
termino_direccion ::= ALFANUMERICO   | DOS_PUNTOS   | DIGITO   | BARRA   | COMA   | RDF   | RESOURCE   | OTRO
```

Figura Nº 18 – Gramática utilizada para parsear las ontologías expresadas en RDFS

Una vez procesada la ontología por el Parser mediante la gramática anterior, se construye una representación intermedia en formato de árbol. La Figura Nº 19 presenta un ejemplo de un árbol creado por el Parser donde se identifican los diferentes conceptos mencionados anteriormente y que será la salida de dicho componente. El Parser y el Recorredor son independientes entre si, ya que se utiliza el subsistema de Persistencia para el almacenamiento de la representación intermedia.

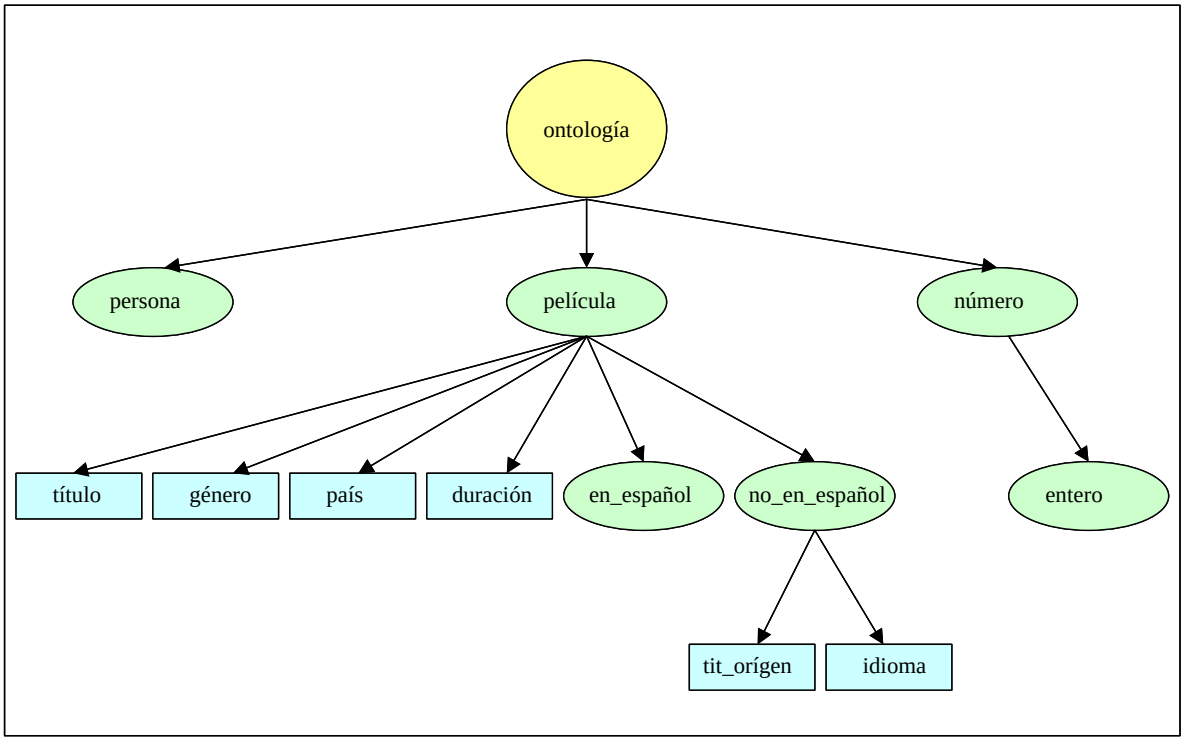


Figura Nº 19 – Ejemplo de árbol de representación intermedia de la ontología

Para la implementación del Parser encargado de aplicar a la ontología la gramática de la Figura Nº 18 y de generar la representación de árbol, se utilizaron las herramientas JLex y CUP, las cuales a partir de una especificación basada en dicha gramática permiten la generación automática del código Java correspondiente al Parser.

Recorredor

El Recorredor es el encargado de generar, a partir de la representación interna de la ontología creada por el Parser, las estructuras de datos que serán utilizadas por el Etiquetador y de almacenarlas utilizando el subsistema de Persistencia.

Está conformado por un Recorredor de Tipos y un Recorredor de Contextos. A continuación explicaremos la función y manera de operar de cada uno de ellos.

Por un lado se hace un recorrido sobre el árbol intentando identificar el conjunto de los tipos que aparecen en las diferentes propiedades definidas en la ontología. Estos tipos están asociados a los valores que tomarán las propiedades de los objetos y mediante el algoritmo de la Figura Nº 20 son colocados en una Lista de Tipos válidos, que será almacenada en el subsistema de Persistencia. Esta lista conforma la salida del Recorredor de Tipos y será utilizada posteriormente por el Etiquetador.

```
Procedimiento Recorrer (Ontologia n)  : Lista de Tipos
     L  =  Lista de Tipos vacia;
     Para cada hijo i de la ontología n
          L = RecorrerAux (i, l )
     Fin Para
     Devuelvo  L
  Fin Procedimiento

Procedimiento RecorrerAux (Descripcion n, Lista de Tipos L) : Lista de Tipos
     Para cada elemento i de la Descripción n
        Si i es una propiedad
          Agrego el tipo de i a L
        Si no
          Si i es una Descripción
                    L = RecorrerAux(i,l);
          Fin Si
        Fin Si
     Devuelvo  L
Fin Procedimiento
```

Figura Nº 20 – Algoritmo del Recorredor de Tipos

Por otro lado, se realiza un recorrido sobre el árbol intentando recolectar los elementos para permitir el análisis de contextos que posteriormente efectuará en Etiquetador. Esta información en organizada, mediante el algoritmo de la Figura Nº 21, en una lista de nodos que será la salida del Recorredor de Contextos y será almacenada en el subsistema de Persistencia. Existe en esta lista, un nodo asociado a cada propiedad identificada de la ontología, y en cada uno de estos nodos se guarda información que permite identificar el concepto al que se aplica la propiedad, el tipo de valores que toma y la lista de palabras claves que tiene asociadas.

```
Procedimiento Recorrer (Ontologia n)  : Lista de Contextos
     L  =  Lista de Tipos vacia;
     Para cada hijo i de la ontología n
          L = RecorrerAux (i, l )
     Fin Para
     Devuelvo  L
Fin Procedimiento

Procedimiento RecorrerAux (Descripcion n, Lista de Contextos L) : Lista de Contextos
     Para cada elemento i de la Descripción n
        Si i es una propiedad
          Agrego el Nodo(Nombre (i), Rango (i), comentario (i)) a L
        Si no
          Si i es una Descripción
               L = RecorrerAux(i,l);
          Fin Si
        Fin Si
     Devuelvo  L
  Fin Procedimiento
```

Figura Nº 21 – Algoritmo del Recorredor de Contextos

4.3.2.2 Estructurador

El módulo Estructurador es el encargado de realizar el pre procesamiento del documento en formato HTML de entrada, dejándolo en un formato conveniente para las siguientes etapas del procesamiento que serán ejecutadas por el Etiquetador y el Generador.

Más concretamente podemos describir las funcionalidades del Estructurador como las siguientes:

- **Identificación de regiones** – Estas regiones, que el Estructurador deberá limitar están asociadas al proceso en el cual se vinculan entre si las instancias ya completamente etiquetadas de elementos de la ontología presentes en el documento de entrada, y que es realizada por el Generador. Esta vinculación se plasma en el documento de metadata que se genera como salida final del Sistema.

- **Etiquetado de oraciones** – Las oraciones representan los límites en los que se basa el proceso de análisis de los contextos de las palabras o expresiones identificadas como instancias de tipos válidos que realiza el Etiquetador.

- **Eliminación de etiquetas html** – Como último punto dentro del cometido de este módulo tenemos la eliminación de las etiquetas propias de un documento en formato HTML que no se hayan utilizado como parte de los dos puntos anteriores.

La implementación de este módulo está basada en transductores que son generados estáticamente, como se explica en el apartado 4.3.1.2, ya que los mismos no dependen de los datos con los que se está trabajando en cada ejecución.

A continuación veremos los principales lineamientos de implementación de cada una de estas funcionalidades.

Identificación de regiones

Estas regiones serán utilizadas por el Generador con la finalidad de verificar la cercanía de dos o más elementos pertenecientes a instancias de un concepto la ontología. Se tomaron dos enfoques diferentes para este punto, los cuales obedecen a decisiones estratégicas del proceso de generación de la metadata:

- El primer enfoque que se tomó fue la **división en párrafos** del documento. Para la identificación de los párrafos se cuenta como único elemento de apoyo, con las etiquetas HTML correspondientes. Este criterio fue evaluado y se encontraron dificultades ya que dichas etiquetas (<P> y </P>) no eran apoyo suficiente para lograr adecuados niveles de precisión (ya que muchas veces en el ambiente Web se utilizan con otra función que la de señalar párrafos) y sobre todo de recall (en los documentos HTML considerados no se utilizaban frecuentemente este tipo de etiquetas).

- Como segundo enfoque, ante el fracaso del anterior y siempre con el objetivo de identificar en el documento regiones que nos sirvieran para indicar la cercanía relativa de varios elementos identificados, se optó por la definición de **zonas delimitadas por los títulos** que aparecían en el documento. En este caso se lograron mejores resultados, sin embargo la variedad de estilos de los documentos considerados hace que varíe el grado de adaptación de este enfoque.

Identificación de títulos

La identificación de títulos realizada por el Estructurador se basa en dos elementos del documento de entrada:

- Las etiquetas HTML de header, nos indican la existencia de un título.
- Reglas específicas, que se basan en el análisis de las palabras escritas total o parcialmente en mayúsculas.

Etiquetado de oraciones

El etiquetado de oraciones es un proceso que se realiza una vez finalizada la identificación de títulos y se compone de las etapas que se muestran en Figura Nº 22 [TRA02]. A lo largo de las diferentes etapas se analiza la aparición de mayúsculas y elementos de puntuación que permitan inferir la presencia de oraciones, para ello de definen reglas en cada etapa las que se verán detalladamente.

La finalidad del marcado de oraciones realizado por este módulo es delimitar el alcance del análisis, que realizará el Etiquetador, de los contextos de instancias identificadas. Esto hace que el etiquetado de oraciones no deba ser tan estricto como si se tratara de un sistema cuya funcionalidad final propiamente la identificación de oraciones. Por lo dicho, en muchos casos de toman simplificaciones en el reconocimiento de oraciones, para evitar complejidad injustificada dentro de este proceso.

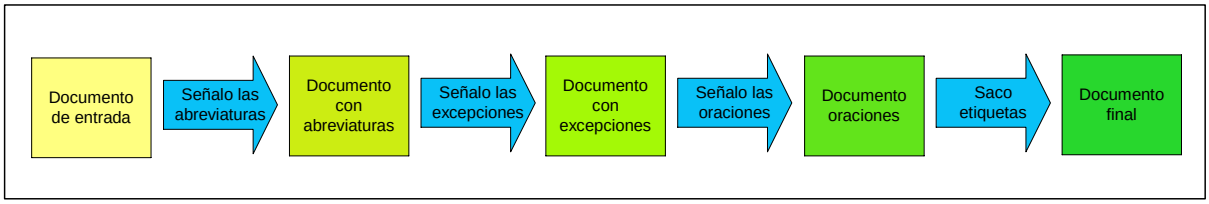


Figura Nº 22 – Etapas del etiquetado de oraciones

A continuación detallaremos cada una de las etapas mostradas en la figura anterior:

- Identificación de abreviaturas – Las abreviaturas representan casos en los que los símbolos de puntuación, claves de este proceso de identificación de oraciones, son utilizados con un sentido diferente al usual. Visto esto, es necesario identificarlas para considerarlas como casos especiales en las etapas siguientes.
- Identificación de excepciones – En esta etapa se identifican símbolos de puntuación que por el contexto en el que se encuentran, no deben ser considerados a la hora de analizar los comienzos de oraciones. Los puntos identificados como parte de las abreviaturas serán considerados como excepciones. La Figura Nº 23 muestra un resumen de las reglas utilizadas por los transductores para la identificación de excepciones de los diferentes signos de puntuación. Es decir que las ocurrencias de dichos signos que aparezcan en los contextos mencionados en la figura, se marcarán como excepciones, y luego no serán tomadas en cuenta en la etapa siguiente cuando se realiza el etiquetado.

Reglas para el punto:

dígito (espacio)* **punto** (espacio)* dígito
punto (espacio)* punto
punto (espacio)* minúscula

Reglas para el signo de interrogación:

interrogación (espacio)* minúscula

Reglas para el signo de exclamación:

exclamación (espacio)* minúscula

Figura Nº 23 – Reglas para el marcado de excepciones

- Marcado de oraciones – En este paso se analizan las instancias de los símbolos de puntuación que no se hayan identificado como excepciones en las etapas anteriores y a partir de determinadas reglas se identifican las oraciones. La Figura Nº 24 muestra un resumen de las reglas utilizadas por los transductores para la identificación de oraciones.

Reglas para el punto:

punto (espacio)* mayúscula
punto (espacio)* dígito
punto (espacio)* paréntesis_abre
punto (espacio)* comilla (espacio)* mayúscula
punto (espacio)* exclamación_abre (espacio)* mayúscula
punto (espacio)* interrogación_abre (espacio)* mayúscula

Reglas para los dos puntos:

dos_puntos (espacio)* mayúscula
dos_puntos (espacio)* dígito
dos_puntos (espacio)* paréntesis_abre
dos_puntos (espacio)* comilla (espacio)* mayúscula
dos_puntos (espacio)* exclabr (espacio)* mayúscula
dos_puntos (espacio)* interrogación_abre (espacio)* mayúscula

Reglas para el signo de exclamación:

exclamación (espacio)* mayúscula
exclamación (espacio)* dígito
exclamación (espacio)* paréntesis_abre
exclamación (espacio)* comilla (espacio)* mayúscula
exclamación (espacio)* exclamación_abre (espacio)* mayúscula
exclamación (espacio)* interrogación_abre (espacio)* mayúscula

Reglas para el signo de interrogación:

interrogación (espacio)* mayúscula
interrogación (espacio)* dígito
interrogación (espacio)* paréntesis_abre
interrogación (espacio)* comilla (espacio)* mayúscula
interrogación (espacio)* exclamación_abre (espacio)* mayúscula
interrogación (espacio)* interrogación_abre (espacio)* mayúscula

Figura Nº 24

- Eliminación de etiquetas – En la etapa final de la identificación de oraciones se eliminan las etiquetas auxiliares que se habían insertado en el texto. Además se analizan el inicio y final del texto para solucionar estos casos particulares.

Eliminación de etiquetas HTML

La tercer tarea que debe realizar el Estructurador es la eliminación de las etiquetas HTML que a esta altura permanezcan en el texto, es decir aquellas que no fueron utilizadas en el proceso de detección de títulos al cual hacíamos referencia anteriormente.

4.3.2.3 Etiquetador

Este módulo tiene como objetivo las acciones referentes a la identificación de instancias de los diferentes elementos que conforman la ontología dentro del texto de salida del Estructurador.

Este proceso se divide en dos etapas (Figura Nº 25). La primera de estas etapas corresponde a la identificación dentro del texto estructurado, de instancias de los dominios identificados como válidos en la ontología ingresada, información que es generada por el Analizador. Estas instancias son "candidatas" a formar parte de la metadata que será generada. Con el fin de confirmar o descartar dicha hipótesis es que se efectúa la segunda etapa del trabajo del Etiquetador. La misma realiza el análisis del contexto de estas instancias en base a la información también provista por el Analizador.

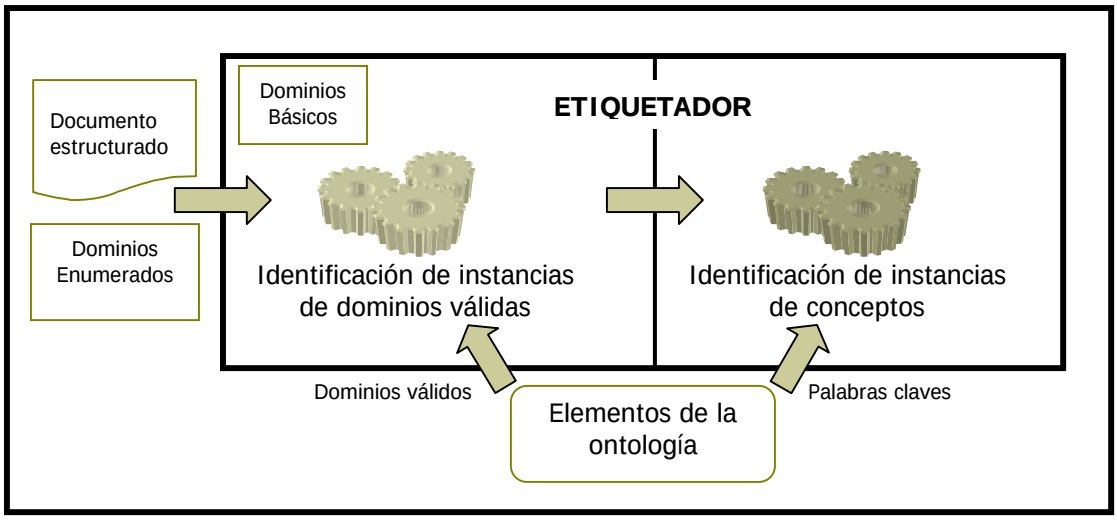


Figura Nº 25 – Etapas asociadas al Etiquetador

A continuación veremos los aspectos más relevantes de cada una de estas etapas.

Identificación de instancias de los dominios válidos

Para cumplir con este objetivo, el Etiquetador toma el documento estructurado (salida del módulo Estructurador) y el listado de los dominios válidos que se generó a partir de la ontología dada por el usuario. Estos dominios válidos se dividen en dos tipos:

- El primero de los tipos, denominados "Tipos Básicos" será evaluado mediante reglas internas del Sistema que se encargarán de la detección de las instancias de unidades, fechas, horas, direcciones web, direcciones de e-mail, números, nombre, lugares y organizaciones. Las reglas para identificar elementos de estos tipos son creadas estáticamente ya que no cambian en función de las entradas, decidiéndose en tiempo de ejecución cuales de ellos y en que orden se aplican al archivo que se está etiquetando.

- El segundo tipo de dominios, cuyos elementos constituyentes son brindados al Sistema por parte de los usuarios, al que llamamos "Tipos Enumerados", comprende a los dominios de los conceptos especificados en la ontología que no se ubican en el punto anterior. En el caso de estos dominios, la especificación de los transductores se realiza en tiempo de ejecución, a partir de los datos provisto por el usuario en los archivos de definición de los tipos enumerados.

Esta clasificación de lugar a otra entrada opcional, que es la localización de algún archivo donde se listen de los elementos que conforman alguno de los dominios enumerados identificados en la ontología.

La identificación de instancias se puede dividir en las siguientes etapas:

- Obtención de archivos de definición de tipos enumerados. En esta etapa el Etiquetador se encarga de obtener los archivos que hayan sido provistos por el usuario para la definición de tipos enumerados.

- Determinación de tipos existentes en la ontología. Se procesa la lista de tipos provista por el Analizador y se intenta identificar cada uno de los tipos pertenecientes a la misma como uno de los tipos conocidos, los llamados tipos básicos. En caso de no ser un tipo básico se lo clasifica como un tipo enumerado.

- Para cada uno de los tipos enumerados identificados en el punto anterior, se genera dinámicamente un transductor que es capaz de identificar instancias del mismo.

- Se aplican los transductores encargados de etiquetar instancias de tipos enumerados, es decir los creados dinámicamente según lo explicado en el punto anterior.

- Se aplican los transductores encargados de etiquetar tipos básicos, en un orden predeterminado, los cuales fueron creados estáticamente.

El seudo código de la Figura Nº 26 resume los cuatro últimos puntos.

```
ListaTipos = lista de los dominios válidos detectados por el
Analizador
Mientras no es vacía ListaTipos hacer
        L = primero(ListaTipos)
        Si L  es un dominio básico entonces
                Agrego L a ListaTiposBasicos
        Si no
                Genero transductor para L (enumerados)
        Fin Si
        ListaTipos = resto(ListaTipos)
Fin Mientras
Aplico transductores para enumerados al documento
Ordeno B
Para cada elemento B de ListaTiposBasicos
        Aplico transductor correspondiente a B al documento
```

Figura Nº 26 – Seudo código de la identificación de instancias de dominios válidos

La tarea de reconocimiento de los diferentes elementos pertenecientes a los dominios válidos se dividió básicamente en seis etapas. Cabe destacar que la complejidad de cada una de estas etapas difiere según el dominio cuyos elementos constituyentes se reconocerán.
Las etapas en las que se dividió el proceso de reconocimiento mostradas en la Figura Nº 27 son:

- Utilización de categorizaciones: en este paso se trata de utilizar la categoría gramatical de cada una de las palabras encontradas en el texto para realizar un relevamiento de cuales de ellas pueden ser instancia o contexto de alguno de los dominios básicos del Sistema. Es en este paso en el que juega un rol importante el uso de una herramienta lingüística como lo es XELDA, facilitando el análisis del texto en este sentido. Como ejemplo de esta etapa, podemos considerar el reconocimiento de fechas. En este caso para evitar identificar instancias de números como una instancia de fecha, se reconocieron en el texto todos los sustantivos y adjetivos obtenidos del procesamiento de XELDA y se descartaron todas las posibles instancias de fechas que se encontraran antecediendo estas palabras. Esto evita que por ejemplo una instancia de número como la que ocurre en la frase “7.526 trabajadores” sea identificada como julio del año 526.

- Uso de diccionarios: este paso se basa en la utilización de un conjunto de palabras que son instancias específicas para un dominio, al cual llamamos diccionario del dominio. Dichas instancias serán evaluadas para determinar si se encuentran contenidas dentro del texto analizado. Asimismo, estos diccionarios también contendrán un conjunto de abreviaciones que se encuentran relacionadas al tipo que se quiere reconocer. Se buscará que los diccionarios generados sean fácilmente extensibles. Como ejemplo de esta etapa, podemos considerar el reconocimiento de tipos enumerados. En este caso, el diccionario estará conformado por una enumeración de las instancias que son válidas para el dominio.

- Establecimiento de elementos disparadores: se consideran elementos disparadores a aquellas palabras o frases que se encuentran inmediatamente antes o inmediatamente después de alguna posible instancia del dominio. En esta etapa se identifican en el texto objetivo aquellas palabras que puedan oficiar de disparadores para el dominio analizado.

- Aplicación de reglas detectadas: en esta etapa se aplican las reglas propias para cada uno de los dominios encontrados, llamando reglas propias a aquellas que definen un patrón o formato específico para el dominio en cuestión. (Por ejemplo dd/mm/aaaa es un formato específico para fechas del cual se deduce una de las reglas para la identificación de este tipo de instancias dentro del texto que está siendo analizado).

- Determinación y corrección de omisiones: esta etapa esta dedicada a la identificación de aquellas reglas que fueron omitidas dentro de las reglas principales, y se enfocan principalmente a aquellos dominios cuyos posibles valores se componen por más de una palabra o por frases cuyas palabras constituyentes pertenecen a varios dominios.

- Detección de marcas que serán eliminadas: cuenta con los elementos necesarios para la eliminación de las marcas escritas en el texto que no se requieren en una evaluación posterior o que representan la identificación incorrecta de los elementos del dominio considerado.

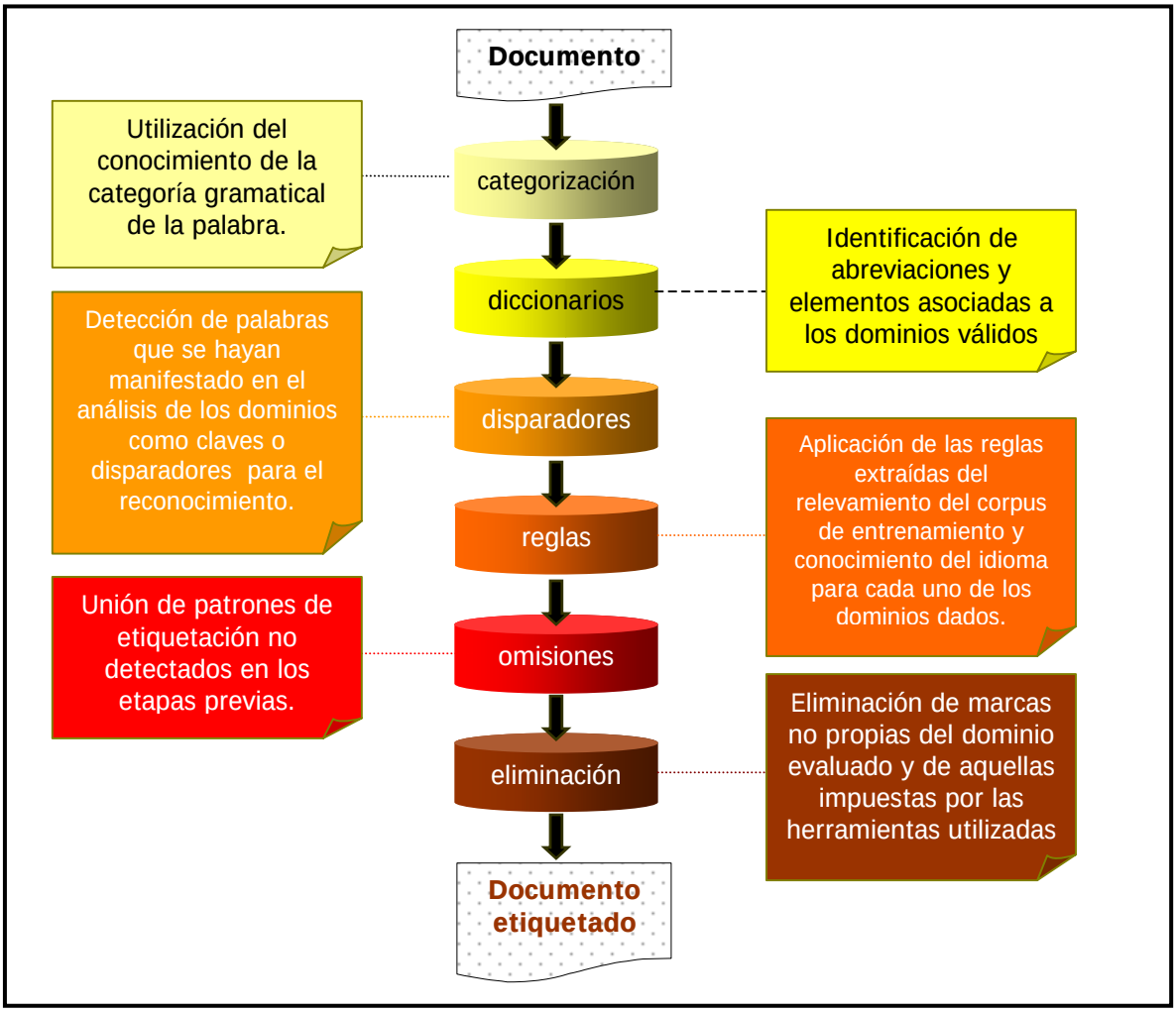


Figura Nº 27 - Etapas del Etiquetador

Como resultado de esta etapa tenemos un documento en el cual se han etiquetado las instancias existentes de los dominios encontrados en la ontología. Esta será la entrada de la segunda parte de análisis realizado por el Etiquetador, la que explicaremos a continuación.

Análisis de contexto

Cada una de las instancias identificadas hasta el momento, es candidata a ser una instancia de un elemento de la ontología. Esta hipótesis será confirmada o descartada en este punto, mediante la realización de un análisis del contexto de la misma.

El criterio que seguiremos consiste en que una instancia de un dominio válido es identificada como una instancia de una propiedad de un elemento de la ontología, si existe antecediendo o precediendo a la instancia, alguna de las palabras claves asociadas a la propiedad, dadas en la ontología ingresada y manipuladas por el Analizador, o si la instancia forma parte de un título dentro del documento.

Para cada una de las propiedades de la ontología, se creará un transductor dinámicamente en función de la información sobre las palabras claves que conformarán el contexto de las instancias. Estos transductores son aplicados al documento salida de la etapa anterior, determinándose así, el documento etiquetado, salida final del Etiquetador.

Cabe destacar que todas aquellas instancias de dominios válidos que no hayan sido identificadas como instancias de alguna propiedad, son descartadas, eliminándose las etiquetas que se le habían colocado en la etapa anterior del procesamiento efectuado por el Etiquetador.

El seudo código de la Figura Nº 28 muestra como el Etiquetador realiza esta parte del análisis.

```
contx = Lista de contextos
Para cada i en contx
        T = creoTransductor(i);
        Agrego T a un script general
Fin Para
Aplico script
Elimino etiquetas de las instancias descartadas
Genero documento de salida del Etiquetador
```

Figura Nº 28 – Seudo código del Análisis de Contextos

Por mayor información en referencia a este módulo consultar el documento asociado al mismo que se adjunta.

4.3.2.4 Generador

La tarea del Generador consiste en procesar el documento etiquetado, es decir aquel en el que se han identificado las instancias de elementos de la ontología, y a partir del mismo generar otro documento conteniendo la metadata asociada al primero, expresada en RDFS.

Para completar la tarea mencionada, este módulo realiza los siguientes pasos:

- En primer lugar realiza una preparación del texto, en la que se eliminan todas las porciones del mismo que no pertenecen a ninguno de los conceptos etiquetados. Este paso deja como resultado un archivo conteniendo líneas, donde cada una de ellas corresponde a uno de los conceptos. Se conserva además la información referente a los tipos, utilizada en el paso siguiente para delimitar zonas.

- Una vez que se ha preparado el archivo, este es procesa por líneas incluyendo cada uno de los conceptos en la metadata de salida, según criterios predefinidos, los cuales utilizan fuertemente la división en zonas del documento propuesta por lo títulos. Se utiliza además parte de la información generada por el Analizador referente a las cardinalidades de las propiedades de la ontología, la cual se encuentra disponible mediante el subsistema de Persistencia.

4.3.3 Verificación

En este punto brindaremos una visión resumida del proceso de verificación que se realizó. Como veremos, la utilización de transductores como herramienta fundamental, introduce en dicho proceso varias particularidades.

Los transductores forman parte de varios de los componentes del Sistema, y hacen que estos no sean totalmente verificables utilizando las técnicas usuales. La razón de esto, es que este tipo de herramientas están asociadas por su naturaleza, a mayores o menores medidas de precisión y recall, dependiendo de su implementación. Estas medidas indican el grado de éxito en el etiquetado que estos transductores logran. Intentar evaluar estos transductores con un enfoque del estilo del que normalmente utiliza un proceso de verificación, en el que por ejemplo se compara la salida con la salida esperada, y se determina el resultado como correcto o incorrecto, reportándose un error en este último caso, no es adecuado ya que como dijimos en el etiquetado siempre se tendrán, en mayo o menor medida, errores.

Por lo dicho, reconocedores como los utilizados por varios módulos del Sistema, los cuales se hayan implementados utilizando transductores, requieren de técnicas específicas de evaluación. Es por esto que a la hora de testear las funcionalidades del Sistema y de cada uno de sus módulos, distinguimos dos casos:

- Por un lado encontramos aquellas funcionalidades resueltas por componentes cuya implementación no involucra máquinas de estado finito. Estas funcionalidades pueden ser verificadas a través de las técnicas usuales.

- Por otra parte encontramos aquellas funcionalidades implementadas en base a transductores. En estos casos, la verificación que es posible realizar utilizando las técnicas usuales es muy limitada, por lo que se complementó con un análisis específico utilizando técnicas para la medición de la precisión y el recall.

Durante la fase inicial del presente proyecto, en el momento en que se planteaban las primeras ideas acerca del proceso de verificación, surgió la idea de incorporar la utilización de la herramienta J-Unit para el manejo de los casos de prueba, buscando dotar al proceso de Verificación de una metodología sólida que le aportara valor agregado al proyecto en su conjunto. Esta herramienta fue evaluada a la vista de la arquitectura del Sistema y se llegó a la conclusión que no se adaptaba bien al prototipo a desarrollar, considerando las características del mismo en cuanto a la utilización de transductores como pieza clave en el proceso de generación de la metadata. Por lo dicho esa idea fue descartada.

En el Documento de Verificación de analizan en detalle tanto la planificación, como la ejecución y resultados de la verificación en cada una de las iteraciones, de forma mucho más detallada.

4.4 Evaluación del prototipo

El objetivo principal de esta sección es presentar una evaluación cuantitativa y cualitativa del prototipo construido. Esta evaluación se concentrará en dos aspectos fundamentales del proceso de generación de la metadata, los cuales se consideran de importancia en el presente proyecto.

El primero de estos aspectos hace referencia a la calidad de la metadata generada. Esta cualidad depende de diversos factores tanto internos como externos. En esta sección buscaremos identificar dichos factores, estudiando el impacto de cada uno de ellos sobre el resultado final. Estos factores pueden ser divididos en tres grandes grupos:

- **Factores externos:** están asociados a la expresividad y completitud de la ontología de dominio y el listado por extensión de los tipos enumerados, y a la estructura que presente el documento de entrada.

- **Factores de diseño de la solución:** relativos al diseño de la solución planteada. Incluyen las decisiones tomadas en referencia a los mecanismos utilizados por los diferentes módulos para conformar la solución global. Se considera a estos factores como internos.

- **Factores referentes a la implementación:** refieren a los detalles de implementación que puedan condicionar de alguna manera la calidad de la metadata, enfocándose a los aspectos de la implementación relacionados con la utilización de máquinas de estado finito. Otros aspectos de la implementación, como el lenguaje de programación utilizado, no impactan sobre la calidad alcanzada en la metadata generada.

Los factores externos y de diseño de la solución serán evaluados con un enfoque global, ya que son considerados como característicos de la solución general.
Por el contrario, los factores de implementación son específicos de cada uno de los componentes del Sistema, por lo que realizaremos un análisis previo por módulos, antes de enfocarnos en el análisis global del prototipo.

Dentro del análisis efectuado a aquellos módulos que se basan principalmente en el uso de herramientas de estado finito para cumplir con sus objetivos, se destaca la utilización de un corpus lingüístico para su evaluación. Un corpus lingüístico es un conjunto de textos almacenados en formato electrónico, y agrupados con el fin de estudiar una lengua o una determinada variedad lingüística. Su objetivo es constituirse en elementos de referencia para el estudio de una frase concreta o un cierto aspecto de una lengua. En la presente evaluación se utilizaron 52 textos (198 páginas) como corpus para el análisis de los transductores involucrados en los diferentes módulos del prototipo divididos de la siguiente manera:

- Se usó un total de 23 textos, constituyéndose un corpus de 112 páginas en la primera iteración, divididos en 13 textos de 53 páginas para la inferencia de reglas y 10 textos con 59 páginas para la evaluación.
- En la segunda iteración se utilizaron 20 textos (99 páginas), divididos en 10 textos con 59 páginas para la inferencia de reglas (que constituían los textos de evaluación de la iteración anterior) y 10 textos con 40 páginas para la evaluación.
- Los 10 textos utilizados para evaluación de la etapa anterior fueron utilizados para la inferencia en la tercera iteración, donde 9 textos con 46 páginas fueron utilizados para la evaluación.

El segundo de los aspectos a evaluar será el tiempo de generación de dicha metadata. Este factor toma gran importancia al tratarse de un prototipo basado en cascadas de transductores, los cuales muchas veces tienen tiempos de ejecución importantes. Cabe destacar que en todos

los casos los tiempos a los que nos referimos son referentes a la ejecución del prototipo en el servidor “Lulu” del Instituto de Computación.

4.4.1 Evaluación de los módulos

Analizador

Este módulo es el encargado de obtener de la ontología los elementos necesarios para el resto del proceso de generación de la metadata.

No existen a nivel de este módulo factores condicionantes de la calidad de la metadata. Esto es consecuencia de que la implementación de este módulo no incluye la utilización de cascadas de transductores. Es por este motivo que la performance de este módulo no afecta el tiempo de ejecución global del Sistema.

Estructurador

Este módulo se encarga de realizar un pre procesamiento del documento de entrada, en función de las necesidades establecidas para el funcionamiento de los demás módulos.

En el caso de este módulo y a diferencia del Analizador, existen factores internos, asociados a la implementación basada en máquinas de estado finito, que afectan tanto la calidad de la metadata generada, como el tiempo de ejecución.

Estos factores se encuentran asociados al etiquetado de oraciones y títulos, que medimos en una primera instancia por los valores cuantitativos de recall y precisión. En el caso de las oraciones, esta medición, dio como resultado:

RECALL	0.902
PRECISION	0.956

De los valores alcanzados, se desprende que en el escenario manejado la mayor dificultad se plantea a nivel del recall que indica la cantidad de ocurrencias identificadas.

Luego de realizar un análisis más detallado sobre el corpus de evaluación, buscando detectar los casos de oraciones que no fueron etiquetadas, encontramos que en la gran mayoría de los casos estas se correspondían con el inicio del texto. Esta peculiaridad se da debido a que generalmente los documentos html no son escritos respetando estrictamente las reglas de puntuación, factor que se intensifica en las regiones del documento correspondientes al inicio del mismo.

Como consecuencia, se produce el etiquetado de varias oraciones como una única. Debido a la ubicación dentro del texto de estas ocurrencias, estas no impactan negativamente en la mayoría de los casos en la metadata generada, ya que las mismas contienen generalmente información adicional de la página, no relevante al dominio de la misma.

En el caso de los títulos se identificaron los siguientes valores de precisión y recall

RECALL	0.724
PRECISION	0.809

La identificación de títulos utiliza fuertemente la aparición de palabras escritas total o parcialmente en mayúsculas. Es por esto que la tarea de identificación de títulos se ve afectada por el hecho de que los documentos Web utilicen las mayúsculas de manera no estricta gramaticalmente hablando. Las carencias en la identificación de títulos afectan de forma importante la generación de metadata, no en la identificación de los elementos constituyentes de los diferentes conceptos, sino que en la generación de la metadata a partir de los elementos

identificados. Dada la utilización de los títulos dentro del Sistema, un bajo nivel de recall da lugar a la disminución de la cantidad de instancias de conceptos de la ontología incluidos en la metadata. Por el contrario, el etiquetado incorrecto de títulos, el cual repercute en la precisión, hace que elementos referidos a una misma instancia de un concepto de una ontología, sean identificados como referentes a instancias diferentes.

Creemos que sería valioso realizar un estudio detallado entorno a este punto, en el cuál se buscara identificar elementos propios del ambiente Web que permitieran mejorar estos niveles de precisión y recall.

Con referencia al tiempo de ejecución, este se encuentra estrechamente relacionado, como cualquier procesamiento con transductores, al largo del documento analizado. En los casos analizados este varía entre 5 y 7 segundos para el largo de los documentos generalmente localizados en la Web.

Este tiempo de ejecución no repercute mayormente en la totalidad del tiempo de ejecución del prototipo, siendo mucho menor al requerido por módulos posteriores.

Etiquetador

Este módulo tiene como finalidad la identificación de instancias de los elementos involucrados en la ontología dada.

Los factores internos de implementación que analizaremos, están asociados, por un lado a la identificación de instancias de dominios válidos, y por otro lado al estudio de los contextos de las instancias previamente identificadas.

Identificación de instancias

En primera instancia nos ocuparemos de analizar los temas vinculados a los transductores reconocedores de instancias, los cuales se dividen en dos grandes grupos.

En el primero de estos grupos encontramos los transductores asociados a la identificación de los llamados tipos básicos: unidades, horas, fechas, números, direcciones web, direcciones de email, nombres, lugares y organizaciones. La siguiente tabla resume los valores hallados de precisión y recall (en el orden previamente citado).

	U	H	F	N	W	E	No	L	O
RECALL	0.96	0.97	0.95	1	1	1	0.78	0.86	0.89
PRECISION	0.93	1	0.78	0.88	1	1	0.9	0.89	0.87

Al contrario del Estructurador, la complejidad de los transductores involucrados en este módulo no aumenta por trabajar en un ambiente Web.

Si bien existen relaciones entre algunos de los reconocedores mencionados (tal es el caso del de números y fechas), cada uno de ellos puede mejorarse independientemente del resto. Mejoras en estos reconocedores, influyen positivamente en la identificación de instancias de dominios válidos.

Si bien se observan variaciones, los niveles de precisión y recall alcanzados permiten una buena identificación de instancias. Uno de los principales obstáculos a la hora de incrementar estos valores, es la ambigüedad inherente del idioma español en relación a los dominios utilizados.

Referente al tiempo de ejecución de los transductores, la siguiente tabla resume la información sobre el tiempo promedio de ejecución de cada uno de los reconocedores, calculados sobre un corpus representativo.

	U	H	F	N	W	E	No	L	O
TIEMPO	1,5s	1,3s	27,8s	2s	1s	1s	1m21s		

Los tiempos de ejecución mostrados en la tabla corresponden al tiempo de ejecución estimado para un documento promedio.

El segundo grupo esta conformado por los tipos enumerados. Para estos tipos, y dado el manejo propuesto por el diseño del prototipo, encontramos que se tiene un recall del 100% ya que no se dejan instancias sin etiquetar. En cuanto a la precisión, podemos decir que la misma no es mejorable en el contexto de la solución propuesta, y esta definida por la ambigüedad propia del idioma en la que una cadena de caracteres que representa una instancia de un tipo puede tener además otros significados.

Respecto a los tiempos de ejecución, encontramos que estos reconocedores se generan dinámicamente en función de una entrada variable dada por el usuario, por lo que cada uno de ellos plantea un tiempo de ejecución diferente. No obstante, dada la forma en la que están construidos, este tiempo varía entre 1 o 2 segundos aproximadamente.

Identificación de contextos

Abordaremos ahora la discusión relativa a los reconocedores involucrados en el análisis de contextos de las instancias de los dominios válidos. El nivel alcanzado por el recall para este tipo de reconocedor es del 100% ya que no se dejan instancias sin etiquetar. En cuanto a la precisión, podemos decir que la misma no es mejorable en el contexto de la solución propuesta, y esta definida por la ambigüedad propia del idioma en la que una cadena de caracteres que representa una instancia de un tipo puede tener además otros significados.

Respecto a los tiempos de ejecución, encontramos que estos reconocedores se generan dinámicamente en función de una entrada variable dada por el usuario, por lo que cada uno de ellos plantea un tiempo de ejecución diferente. No obstante, dada la forma en la que están construidos, este tiempo varía entre 1 o 2 segundos aproximadamente.

Generador

Este módulo cumple con el cometido de generar la metadata asociada al documento etiquetado.

No existen a nivel de este módulo factores condicionantes de la calidad de la metadata. Esto es consecuencia de que la implementación de este módulo no incluye la utilización de cascadas de transductores. Es por este motivo que la performance de este módulo no afecta el tiempo de ejecución global del Sistema.

4.4.2 Evaluación del Sistema

En este punto se analizaran a nivel global los diferentes aspectos que influyen tanto en la calidad de la metadata generada, como en el tiempo de generación.

Calidad de la metadata

Los aspectos que influyen sobre la calidad de la metadata generada abarcan desde aspectos de diseño de la solución, hasta la diversidad de características presentadas por los documentos que conforman el flujo de entrada del Sistema.

Factores externos

Dentro de los aspectos referidos a los documentos de entrada del Sistema podemos distinguir los siguientes puntos:

- Ontologías

La riqueza de la metadata generada esta fuertemente condicionada al grado de expresividad de las descripciones de dominio utilizadas, es decir a cuan completas sean estas a la hora de describir el dominio de acción, característica que se ve afectada por la falta actualmente de una metodología estándar que facilite su creación con buenas propiedades.

Enfocándonos dentro del proyecto, uno de los elementos fundamentales para que la ontología pueda ser totalmente aprovechada, es la riqueza en cuanto a las palabras claves que sirven para identificar los contextos asociados a los distintos elementos.

A partir de una ontología que no cuente con las características de expresividad mencionadas, no es posible generar una metadata semánticamente completa.

- Definición de tipos enumerados

La completitud de la definición de tipos enumerados influye en la identificación de los mismos en el documento. La omisión de valores posibles en la definición de estos tipos, hace que las ocurrencias de estos valores en el documento que se esta procesando no sean identificados y por tanto no conformen la metadata generada.

- Documento a etiquetar

Como veremos a continuación, existen determinados formatos de documentos cuya estructura se adapta mejor a la lógica de la solución propuesta. El formato del documento está relacionado a la distribución de la información dentro del mismo, separada por secciones, títulos, correcta utilización de los signos de puntuación, etc.

Factores de diseño

A continuación veremos como cada una de las decisiones tomadas en el diseño de la solución afecta la generación de la metadata. La solución es la descrita en la Sección 3.4.

- Utilización de una ontología como descripción de un dominio

La introducción de un modelo conceptual del dominio de los documentos a ser analizados, trae consigo las ventajas asociadas a la estandarización de los conceptos pertenecientes al dominio de acción. Una descripción abarca todo el espectro de documentos pertenecientes al dominio

que describe. Este enfoque, utilizado por la solución propuesta, permite tener un alto grado de independencia con los elementos asociados a la presentación del documento, posibilitando esto que el proceso de generación de metadata no este asociado a esta presentación. La representación del modelo conceptual mediante una ontología tiene como ventaja la existencia de un gran número de lenguajes de especificación, lo que abre una amplia gama de posibilidades a la hora de optar por aquel que brinde la flexibilidad y expresividad buscadas.

El uso dado por lo solución a la ontología, permitió concentrar en un solo paso previo a la generación de la metadata, la intervención requerida de un usuario. Esto posibilita que un usuario sin conocimiento del dominio sea capaz de generar metadata de forma totalmente automática a partir de la definición de la ontología realizada previamente.

Por lo dicho, consideramos que la decisión de utilizar una ontología como descripción del dominio analizado fue un valioso aporte en favor del enriquecimiento del valor de la calidad de la metadata generada.

 Elección de lenguaje de especificación de la ontología

RDFS, lenguaje elegido para la especificación de la ontología, otorgó la flexibilidad necesaria para expresar, manipular y extraer de forma sencilla la información requerida para el correcto funcionamiento de la solución.

El hecho de que RDFS se proyecte como un lenguaje estándar para la descripción de dominios, hace que las ontologías requeridas sean de fácil obtención y mejor calidad.

 Ubicación de la información referente a palabras claves

La utilización de las etiquetas comment para la especificación de las palabras claves para la identificación de instancias de los conceptos asociados a la ontología, permitió facilidad en la extracción y manipulación de las mismas. Asimismo, dotó a la solución de una gran potencia a la hora de identificar en el texto estas instancias.

Por otro lado, el requerimiento del uso particular de la etiqueta en este sentido hace que se requiera un mayor grado de especificidad de la ontología dada. Esto si bien es un requerimiento adicional, es una de las bases de la automatización lograda en el resto del procesamiento.

 Ubicación de los tipos enumerados

La solución contempla la utilización de diferentes tipos de repositorios para la especificación de las instancias validas de estos tipos. Esto dota a la solución de una gran potencia para el aprovechamiento de información que pudiera existir sobre este tipo de dominio, mayormente pensando en la utilización de la solución en un ambiente Web.

Esta información se presenta como un complemento a la dada por la ontología, lo cual implica un requerimiento adicional de la solución que traerá como ventaja la ampliación del espectro de los conceptos de la ontología que es posible identificar en los documentos.

 Representación interna de la ontología

La utilización de árboles para la representación interna de la ontología facilitó la manipulación de los elementos de ella extraídos. La transformación de la ontología a esta representación interna se realiza en forma natural, respetando la estructura jerárquica implícita en la misma.

- Pre procesamiento del documento

El hecho de haber incluido una instancia de pre procesamiento del documento de entrada que contemple las necesidades de los demás módulos, posibilitó obtener una entrada más apropiada a los mismos, eliminando la información asociada a la presentación que es característica de los documentos presentes en el escenario Web, la cual dificulta la labor de los otros módulos.

- Límites en la búsqueda de contextos

Una de las decisiones críticas de diseño de la solución propuesta, fue el hecho de determinar el alcance en la exploración de los contextos de aquellas palabras que hayan sido identificadas como candidatas a ser instancias de los conceptos de la ontología. El análisis de contextos se basa en la división en oraciones del documento. Referente a este punto, surgen diferentes alternativas que contemplan soluciones basadas en la cantidad de oraciones contiguas incluidas en dicho contexto.

Un análisis efectuado sobre un corpus de inferencia, permitió detectar que las ocurrencias de palabras claves se manifiestan en la cercanía a la palabra a la que están referidas. Esto se corresponde con propiedades del lenguaje natural que hacen referencia a la cercanía dentro de un texto de palabras relacionadas con un mismo concepto. Es por esto que finalmente se optó por limitar el alcance de estos contextos a la misma oración en la que aparece el término.

Como evaluación del enfoque tomado encontramos que a pesar de la dificultad adicional de trabajar con documentos Web, que no trabajan con estructuras gramaticalmente completas, se logró un buen nivel de éxito.

- Límites de regiones

El proceso de vinculación de diferentes elementos de la ontología detectados en el documento esta basado en la división en regiones de dicho documento. Estas regiones representan porciones del documento que contienen información referente a una misma instancia de un elemento de la ontología.

Esta tarea debe contemplar varios aspectos. El primero de ellos esta dado por la definición de los límites de estas regiones, y el segundo esta asociado a las decisiones tomadas en el caso de que existan regiones que contengan información sobre más de una instancia, o que información de una misma instancia se encuentre localizada en más de una región.

Esta vinculación de instancias es una tarea compleja en la medida en que la información en el documento no se encuentre distribuida en una forma acorde a la organización del mismo, ya que dicha organización representa los únicos elementos disponibles como base para este proceso.

Con esto queremos decir que existen diferentes tipos de documentos, algunos de los cuales se ajustan mejor a la definición de regiones que mencionamos. Documentos organizados en secciones, ricos en títulos y subtítulos, son más propicios a obtener mejores resultados mediante este enfoque debido a que se reduce la cantidad de casos en los que la información se encuentra diseminada en diferentes regiones, mientras que otros documentos sin estas características, donde la información se encuentra desordenada, requieren mayor análisis, implicando una mayor dificultad.

Aún en documentos organizados en secciones, encontramos diferentes tipos en cuanto a como se distribuye la información en el documento. Los mejores resultados obtenidos se dan en documentos de tipo catálogo, listados, o estructuras similares, siendo estos el tipo de documentos en los que enfocamos mayormente nuestro trabajo.

En cuanto a la definición de las regiones se evaluaron e implementaron dos alternativas. La primera de ellas basada en la división en párrafos y la segunda en la utilización de títulos como delimitadores de las regiones.

La primera de estas opciones presentó varios inconvenientes, por un lado la dificultad de identificación de párrafos causada por la utilización de documentos Web, y por otro lado la dispersión de los conceptos en regiones contiguas.

A causa de esto se optó por el segundo enfoque, asociado a la utilización de títulos, elementos encontrados comúnmente en documentos del ambiente Web, que permitió mejorar la consolidación de la información de conceptos de la ontología.

- Elección de lenguaje de especificación de la metadata

La elección de RDFS como lenguaje para especificación de la metadata permitió, dada la información identificada, escribir de forma natural las instancias localizadas. Asimismo, al ser un lenguaje estándar permite el cómodo aprovechamiento de la metadata generada para diversos fines, siendo útil para una amplia gama de usuarios.

Tiempo de ejecución

Las acciones tomadas a nivel de diseño e implementación con el objetivo de disminuir el tiempo de ejecución de las máquinas de estado finito involucradas, hicieron que el mismo se encontrará dentro de parámetros razonables y fuese mucho menor al que hubiese tenido asociado una implementación sin estas medidas.

Los resultados en cuanto al tiempo de ejecución global del Sistema que podemos mencionar, se desprenden directamente de los resultados del análisis por componentes que efectuamos en el punto anterior. Para un documento dado, aproximadamente el 94.5% del tiempo de ejecución corresponde al módulo Etiquetador, 5.5% se identifica con las tareas realizadas por el Estructurador, siendo el tiempo ocupado por el Analizador y el Generador despreciable.

Para un documento típico de 2 carillas, el tiempo insumido es de aproximadamente 2 minutos 30 segundos, mientras que para un documento más extenso de alrededor de 20 carillas, el tiempo es de alrededor de 12 minutos (tiempos medidos sobre el servidor “Lulu” del INCO).

Capítulo Nº 5: Conclusiones

En este capítulo, analizaremos las conclusiones obtenidas como resultado del presente proyecto.

Las mismas las dividimos en tres grupos, de acuerdo a su alcance. Como veremos, algunas de ellas corresponden al problema general de extracción de metadata de documentos Web, otras a la aplicabilidad de las Técnicas de Estado Finito para atacar dicho problema y las restantes a la solución particular propuesta como parte de este proyecto.

Cada una de las secciones siguientes trata las conclusiones referentes a cada uno de estos grupos.

5.1 Sobre el problema de generación automática de metadata en un escenario Web

Las herramientas que existen en la actualidad para resolver el problema de extracción de información se basan en un procesamiento semiautomático, fuertemente ligado a la interacción con el usuario, quien a través de esta interacción aporta información básica para la extracción.

La completa automatización de este proceso representa un desafío importante, ya que no se cuenta con la información brindada por los usuarios, y se tienen como única fuente de conocimiento las entradas asociadas al proceso. La calidad de la información contenida en estas entradas es generalmente menor a la de la información que un usuario puede aportar al proceso.

Otro elemento, propio de los problemas de extracción de información, que dificulta esta tarea, es el alto grado de ambigüedad inherente que presenta el lenguaje natural. Este problema se ve agravado en un escenario Web, donde muchas veces no se siguen de manera estricta las reglas gramaticales propias del lenguaje. Además, la diversidad de estructuras de los documentos y los dominios de información existentes en el ambiente Web, introducen particularidades a la hora de tratar de resolver el problema de extracción de información en este ambiente.

Dados estos elementos, el problema se presenta como complejo, siendo necesario recurrir al aumento de la expresividad de las entradas en favor a mejorar la calidad del procesamiento automático.

5.2 Sobre las técnicas de estado finito

En este proyecto, se apuntó a la factibilidad de resolver el problema anteriormente mencionado mediante la utilización de técnicas de estado finito. El diseño de una solución e implementación de un prototipo basado en estas técnicas permitió observar ciertas características del comportamiento de las mismas aplicadas a este problema.

Las técnicas de estado finito empleadas mostraron un alto grado de flexibilidad a la hora de procesar el texto buscando identificar información relevante dentro del mismo. La razón de esta afirmación, es la facilidad encontrada para realizar el pasaje de las reglas detectadas para reconocer distintos elementos a su especificación como máquinas de estado finito.

Por el contrario, las técnicas no presentan facilidades a la hora de consolidar la información identificada en función de una descripción de un dominio particular, efectuando un mapeo entre ellos. Esta cualidad es propia a la naturaleza de este tipo de técnicas, ya que el objetivo de las mismas se centra en el reconocimiento de elementos lingüísticos basado en reglas específicas.

Otra característica de las máquinas de estado finito, esta relacionada con el tiempo de ejecución de las mismas. Este está estrechamente relacionado con el diseño asociado a su construcción y el largo del documento analizado, y de no tomarse las medidas necesarias, este tiempo podría transformarse en una limitante para el uso de estas técnicas.

Haciendo un balance de los elementos mencionados, consideramos que las técnicas de estado finito son aplicables al problema de extracción de metadata y representan un aporte valioso en la tarea de reconocimiento de información relevante dentro de un documento en base a una especificación, siendo necesario complementar este tipo de herramientas con elementos que permitan resolver aspectos referentes a la posterior consolidación de dicha información.

5.3 Sobre la solución propuesta

En el presente proyecto se diseñó e implementó un prototipo para la resolución del problema de extracción automática de metadata basado en técnicas de estado finito. Esta solución se construyó sobre la base del análisis efectuado durante las primeras etapas del proyecto, y tuvo como pilar fundamental para la automatización del proceso la búsqueda de ciertas propiedades en la descripción de dominio ingresada.

La solución planteada, logra buenos resultados en la etapa de identificación de instancias de elementos de la ontología dentro del documento, mientras que en la etapa de consolidación de esta información para la generación de la metadata se comporta bien para documentos con determinadas características en cuanto a su estructura y organización de la información como son los catálogos o listados. Para otros tipos de documentos, se requiere continuar el estudio de un método de adaptación de la presente solución en busca de obtener los mejores resultados posibles.

Con referencia a la calidad de la metadata generada, podemos decir que se obtuvieron buenos resultados, dada una descripción de dominio lo suficientemente expresiva y que contemple los rasgos especiales requeridos (como la utilización de palabras claves para el contexto) y un documento con las características mencionadas anteriormente.

La solución contempla dentro del diseño de los transductores el tema del tiempo de ejecución, manteniendo el mismo dentro de parámetros aceptables.

Otro de los rasgos a destacar, es la correcta integración de las herramientas utilizadas como parte de esta solución.

El modelo de proceso elegido para el desarrollo de esta solución resultó adecuado, ya que brindó las facilidades necesarias para la mejora continua de los diferentes módulos en función de la aplicación de los resultados obtenidos en evaluaciones intermedias.

Por sus características, la solución planteada abre la posibilidad de trabajar en dirección a la extensión de la misma y mejora de sus componentes. Los resultados obtenidos hasta el momento son alentadores para seguir trabajando en la dirección propuesta, en la que creemos que con alta probabilidad se continuarán logrando mejoras en referencia a la solución del problema planteado.

5.4 Aportes del proyecto

Este proyecto ofrece una aproximación a la solución de un problema complejo y de interés en el área de aplicación, ya que refiere a una temática de actualidad de amplia proyección.

La solución representa un enfoque novedoso en el estudio de dicho problema y el análisis de los resultados obtenidos permite identificar aquellos factores en los que se encuentran las principales dificultades.

Además, establece los elementos para la continuidad del trabajo en esta dirección, o en otra que tome las mejores características de la solución propuesta.

Presenta una aplicación concreta de las técnicas de estado finito para la resolución del problema dado, obteniéndose un prototipo final que implementa la solución descrita.

5.5 Trabajo futuro

Como sugerimos anteriormente, existen numerosos aspectos en los que se puede trabajar en pro de mejorar la solución propuesta.

En cuanto a incorporar mejoras dentro del alcance actual de la solución, podemos mencionar los siguientes aspectos:

- Incorporación de herramientas lingüísticas que realicen un estudio de sinónimos (diccionarios de sinónimos) y de los tiempos verbales permitiendo lograr un mayor aprovechamiento de los elementos componentes de la información referente a los contextos encontrada en la ontología.
- Incorporación de procedimientos que permitan el acceso a diferentes tipos de repositorio con el fin de obtener información de los tipos enumerados
- Continuar mejorando los niveles de recall y precisión de cada uno de los reconocedores involucrados en la solución general.
- Continuar mejorando la performance de los transductores involucrados, buscando mejorar los tiempos de procesamiento.
- Continuar mejorando las heurísticas para la consolidación de la metadata una vez identificadas las instancias dentro del documento.

Referente a la ampliación del alcance de la solución actual encontramos los siguientes puntos:

- Buscar formas de adaptación o modificaciones necesarias en la solución actual para su eficaz aplicabilidad a otros tipos de documentos
- Integración de módulos que permitan la utilización de ontologías expresadas en otros lenguajes.
- Integración de módulos que permitan generar la salida en diferentes lenguajes.
- Ampliación de los dominios básicos reconocidos por la aplicación.

5.6 Reflexiones finales

Creemos que los objetivos del proyecto fueron cumplidos en buena forma, ya que no solo se obtuvo una implementación concreta de una primera aproximación a la solución del problema, sino que además se realizó una evaluación detallada de los diferentes elementos que agregan complejidad al mismo.

Las conclusiones a las que se llegaron dejan sentadas las bases para la continuidad de la investigación bajo estos lineamientos.

REFERENCIAS

[BEE99] K. Beesley y L. Karttunen. Finite-State Morphology: Xerox Tools and Techniques. CSLI Publications, Stanford, CA. 999.

[BRO00] J. Broekstra, M. Klein, S. Decker, D. Fensel, I. Horrocks. Adding formal semantics to the Web building on top of RDF Schema. In Proc. SemWeb 2000, 2000.

[CUP] http://www.cs.princeton.edu/~appel/modern/java/CUP/
Ultima visita: 25 de febrero de 2004.

[EXC] www.hummingbird.com/products/nc/exceed
Ultima visita: 25 de febrero de 2004.

[FRA96] S. Franklin, A. Graesser. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Institute for Intelligent Systems. University of Memphis. 1996.

[HEN99] J. Hendler. Is There an Intelligent Agent in Your Future? Marzo 1999.

[HOP79] J. Hopcroft and J. Ullman. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Massachusetts, 1979.

[IAN97] R. Iannella, A. Waugh. Metadata: Enabling the Internet. CAUSE 97. 1997.

[JAC] http://www.rational.com/products/rup/index.jsp
Última visita: 25 de febrero de 2004.

[JAV] http://java.sun.com/
Última visita: 25 de febrero de 2004.

[JBU] www.xwin32.dk/
Última visita: 25 de febrero de 2004.

[JLE] www.cs.princeton.edu/~appel/modern/java/Jlex/
Última visita: 25 de febrero de 2004.

[KAR94] L. Karttunen. The Replace Operator. In 33th Annual Meeting of the Association for Computational Linguistics, M.I.T. CambridgeMass. 1994.

[KUH02] S. Kuhlins, R. Tredwell. Toolkits for Generating Wrappers – A Survey of Software Toolkits for Automated Data Extraction from Websites. Net.ObjectDays 2002: Objects, Components, Architectures, Services and Applications for a Networked World. 2002.

[LAE02] A.H.F. Laender, B.A. Ribeiro-Neto, A.S. da Silva and J.S. Teixeira. A Brief Survey of Web Data Extraction Tools. *SIGMOD Record*, Vol. 31, No. 2, pp. 84-93, June 2002.

[MOU01] A. Carvalho Moura. A Web Semântica: Fundamentos e Tecnologías. VI Congreso Internacional de Ciencias de la Computación, La Paz, pp. 46-82, out. 2001.

[MUS99] I. Muslea. Extraction Patterns for Information Extraction Tasks: A Survey. The AAAI-99 Workshop on Machine Learning for Information Extraction, Technical Report WS-99-11, Orlando, Florida, 19 July 1999.

[NOY01] N. F. Noy, D. MacGuiness. Ontology development 101: A guide to creating your first ontology – Standford Knowledge Systems Laboratory. 2001.

[RDF] http://www.w3.org/RDF/
Última visita: 25 de febrero de 2004.

[RDFS] http://www.w3.org/TR/2004/REC-rdf-schema-20040210/
Última visita: 25 de febrero de 2004.

[SEI00] R. S. Seiner. Questions metadata can answer. Computer Associates Products, CAI, 14 de julio de 2000.

[SSH] http://www.ssh.com/
Última visita: 25 de febrero de 2004.

[TRA02] D. Calegari, S. González, M. Ifrán. Proyecto final de la materia "Transductores" Curso 2002, Grupo 05. Facultad de Ingeniería – Universidad de la República.

[VEN] S. Venkatasubramani, Ravan RKVS. Annotations in Semantic Web. National Center for Software technology. India.

[WS] http://www.semanticweb.org/
Última visita: 25 de febrero de 2004.

[XEL] http://www.xrce.xerox.com/competencies/past-projects/platforms/xelda.html
Última visita: 25 de febrero de 2004.

[XFS] http://www.xrce.xerox.com/competencies/content-analysis/fst/home.en.html
Última visita: 25 de febrero de 2004.

[XWI] www.xwin32.dk/
Última visita: 25 de febrero de 2004.

BIBLIOGRAFÍA

Regular Models of Phonological Rule Systems. R. Kaplan – M. Kay (1994) Computational Linguistics 20:3 331-3378. 1994

FASTUS, a Cascaded Finite-State Transducer for Extracting Information from Natural Language Text, J.R.Hobbs et al. in Finite-State Language Processing, E. Roche, Y. Schabes eds., MIT Press 1997

Jedi: Extracting and Synthesizing Information from the Web.Gerald Huck, Peter Fankhauser, Karl Aberer, Erich Neuhold:In: Michael Halper (Ed.), Proc. 3rd IFCIS Intl. Conf. on Cooperative Information Systems, CoopIS'98, New York City, New York, USA, August 20-22, 1998. Los Alamitos, California, IEEE Computer Society Press, 1998, pp. 32-43, ISBN 0-8186-8380-5

Example Based Wrapper Generation.Nitesh Shrestha, Ralph Busse, Gerald Huck. Editors: Behrouz Homayoun Far, M. Hassan Shafazand, Makota Takizawa, Roland Wagner. Conference: EurAsia-ICT 2002, Advances in Information and Communication Technology, First Eurasian Conference , Shiraz, Iran, October 2002, Workshop Proceedings. Publisher: Oesterreichische Computer Gesellschaft, Austrian Computer Society Page no:493-500, ISBN 3-85403-161-0 Oesterreichische Computer Gesellschaft