Lavinia: Un entorno para Procesamiento de Lenguaje Natural

Cecilia Techera[1], **Diego Garat**[1], **Guillermo Moncecchi**[1 2]

[1] Instituto de Computación - Facultad de Ingeniería - Universidad de la República
Montevideo, Uruguay

[2]Laboratoire MoDyCo, UMR 7114 CNRS - Université Paris Ouest
Nanterre La Défense, France

{ctechera,dgarat,gmonce}@fing.edu.uy

Abstract. *En este artículo presentamos Lavinia, un ambiente para Procesamiento de Lenguaje Natural (PLN), en donde desarrolladores y usuarios pueden integrar y compartir componentes construidos en la plataforma UIMA. Lavinia introduce un algoritmo para visualizar los resultados independiente del proceso que los generó, permitiendo además al usuario modificar en forma dinámica la forma en que se muestran, buscando destacar aquellos aspectos del análisis que sean de su interés.*

1. Introducción

Por lo general, las aplicaciones de Procesamiento de Lenguaje Natural (PLN) se implementan como un conjunto de tareas pequeñas organizadas en un flujo: la salida de una tarea es la entrada de la siguiente.Gran parte del esfuerzo de construir una aplicación de PLN no radica en la implementación de los módulos requeridos —la mayoría pueden encontrarse como software de código abierto—, sino en lograr que estos diferentes módulos se comuniquen e intercambien datos. Teniendo en cuenta este problema, la comunidad de PLN ha estado trabajando en enfoques para aumentar la interoperabilidad de las soluciones, de modo de facilitar su integración en nuevas aplicaciones. Esto incluye la definición de estándares de representación de objetos de análisis, la configuración de las diferentes tareas y la construcción del flujo de ejecución [5, 2, 1].

Lavinia[1] nace con el objetivo de construir un ambiente web que ayude en el proceso de integración y ejecución de módulos reutilizables. El núcleo de Lavinia es la plataforma UIMA[3], y sobre él se agregan mecanismos para facilitar su uso, posibilitando: (a) generar, dinámicamente, flujos formados por componentes previamente disponibles ; (b) guardar y cargar su configuración; (c) exportar los resultados de análisis intermedios o finales a un archivo; y (d) visualizar los resultados de análisis de acuerdo a opciones que elige el usuario. La arquitectura de la aplicación se muestra esquemáticamente en la figura 1

1.1. El núcleo de procesamiento: UIMA

La arquitectura que seleccionamos como núcleo para el procesamiento fue UIMA (Unstructured Information Management Architecture) [3], una plataforma libre para la creación, integración y desarrollo de aplicaciones de gran escala que manejan información no estructurada, cuyo objetivo principal es proveer un entorno común en donde

[1]Lavinia está disponible en http://www.fing.edu.uy/inco/grupos/pln/lavinia.html

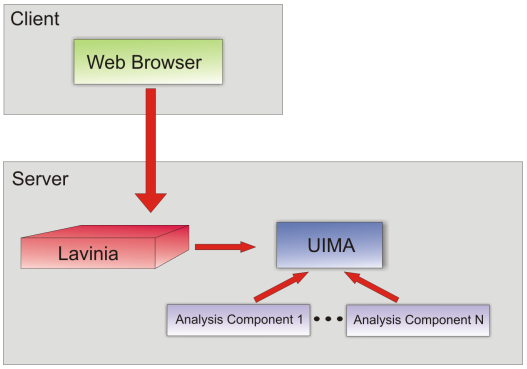

Figura 1. Esquema de Lavinia

los desarrolladores puedan colaborar en la creación e intercambio de soluciones de PLN, aprendizaje automático y recuperación de información.

Como núcleo de procesamiento, UIMA provee a Lavinia de un formato único de representación de anotaciones referidas a textos (CAS, *Common Analysis System*), de un sistema de tipos jerárquico y con atributos asociados para las anotaciones, y de la infraestructura para la definición de módulos de análisis y su ejecución en flujo.

1.2. Mejora de experiencia del usuario

Teniendo a UIMA como un núcleo sólido que da acceso a toda la metadata que representa resultados de análisis, nos enfocamos en la experiencia del usuario.

Como primer objetivo intentamos construir una solución que, accediendo a una página web, permita la selección de diferentes módulos de análisis de texto, construcción del flujo de componentes deseado y configuración de parámetros y recursos de los módulos. Cuando se introduce un componente en un flujo, Lavinia muestra sus tipos de entrada y salida (además de los parámetros disponibles), y verifica, en tiempo de ejecución, si el flujo está bien construido, esto es, si la salida de los componentes previos en la cadena contiene la metadata necesaria para el análisis de cada componente.

Desde la perspectiva de diseño de componentes, Lavinia permite la rápida integración de nuevos módulos de análisis: es necesario construir un módulo compatible con UIMA, subirlo utilizando la interfaz web y declarar su sistema de tipos, parámetros y recursos externos requeridos (diccionarios, archivos de configuración, etc.). A partir de ese momento, el módulo estará disponible para todos los usuarios de Lavinia, los cuales podrán utilizarlo en sus flujos de análisis.

Para el diseño de la interfaz optamos por el uso de interfaces web dinámicas basadas en comunicaciones asíncronas y clientes livianos, utilizando tecnología Ajax, con el objetivo de facilitar su uso por parte de usuarios no especializados en software, aunque es un aspecto pendiente en nuestro trabajo el medir cuánto contribuyó esta decisión en lograr este objetivo.

1.3. Visualización de resultados de análisis

Un aspecto que consideramos destacable de Lavinia es la posibilidad de visualizar en un formato uniforme los resultados de análisis, permitiendo cambios en su visualización para destacar los aspectos del análisis más importantes para el usuario. En UIMA,

```
Para cada tipo de anotación ti, en orden de prioridad
   Para cada anotación aj, del tipo ti
      Para cada celda entra aj.inicio y aj.fin
          Si la celda tiene color de fondo
            Si la celda tiene bordes
               Asignar borde a aj.inicio y aj.fin
            sino
               Asignar bordes
          sino
            Asignar color de fondo
       fin
    fin
fin
```

Figura 2. Pseudocódigo del algoritmo de visualización

los resultados de análisis son anotaciones sobre el texto con marcas de inicio y fin, y una serie de atributos que pueden incluso referir a otras anotaciones del texto. Para mostrar esos resultados, trabajamos sobre la filosofía de que es el usuario, y no la aplicación, quien decide las anotaciones a visualizar y cómo se mostrarán.

Desarrollamos, entonces, un mecanismo para visualizar las anotaciones, el cual se encuentra en el módulo de visualización de Lavinia. Este mecanismo se basa en capas de anotaciones con diferentes prioridades, de manera similar a los típicamente utilizados en los Sistemas de Información Geográfica. Cada tipo de anotación generado por un flujo de análisis es una capa, y el usuario puede asignarle un orden de preferencia, si será visualizada o no, y en caso afirmativo, los colores de fondo y tipo de letra a utilizar.

El módulo de visualización muestra las capas según el orden de prioridad. Cuando una anotación se solapa con un segmento de texto que ya fue pintado —en otras palabras, parte de la anotación ya fue pintada por la existencia de otra anotación con mayor prioridad—, el módulo de visualización sólo pinta un borde alrededor del segmento correspondiente, sin rellenarlo. Entonces, si dos anotaciones se solapan, la que tiene mayor prioridad se mostrará sobre la otra. En el caso en que ya se tienen dos anotaciones solapadas —esto es, el segmento tiene un color de fondo dado por la primer anotación, y un color de borde, dado por la segunda anotación—, la tercer capa sólo se muestra como un par de *brackets* al principio y al final del segmento. El pseudocódigo del algoritmo de visualización se muestra en la figura 1.3.

Todos los atributos de la capa pueden ser modificados dinámicamente luego de realizar el análisis; la pantalla de visualización será actualizada sin necesidad de tener que realizar el análisis nuevamente. De esta forma, el usuario puede ver rápidamente los resultados de muchas maneras distintas permitiendo destacar distintos fenómenos lingüísticos. Adicionalmente, al seleccionar una porción del texto, Lavinia mostrará todos los atributos asociados a sus anotaciones. Las figura 3 muestra cómo la visión global de la salida de un flujo puede ser drásticamente modificada. En la figura se muestra a la izquierda la configuración de colores, qué anotaciones se mostrarán y sus prioridades).

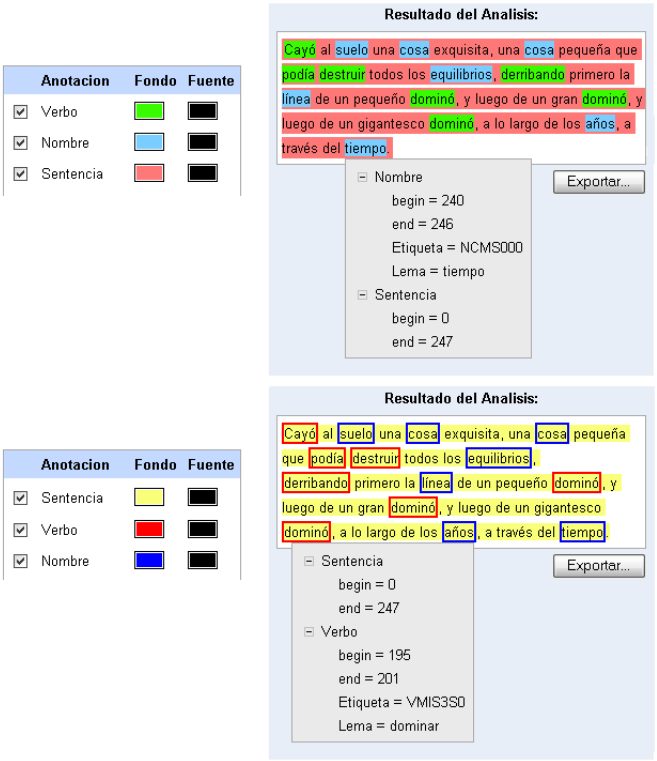

Figura 3. Resultados de un análisis visualizado de dos formas distintas

2. Conclusiones

Lavinia es un ambiente que permite a personas interesadas en análisis y procesamiento de lenguaje natural acercarse a las herramientas del área, aun sin tener conocimientos de programación, y poder visualizar de forma uniforme y amigable los resultados de sus análisis. Por otra parte, agrega una nueva perspectiva a los desarrolladores de PLN: no necesitan instalar y configurar la plataforma ni las herramientas que requiere; en su lugar pueden simplemente utilizar los módulos de análisis en un navegador y verificar los resultados. De esta forma, nuestra plataforma web ayuda a realizar prototipos de aplicaciones más complejas.

Referencias

[1] Steven Bird, David Day, John S. Garofolo, John Henderson, Christophe Laprun, and Mark Liberman. Atlas: A flexible and extensible architecture for linguistic annotation. *CoRR*, cs.CL/0007022, 2000.

[2] Steven Bird and Mark Liberman. A formal framework for linguistic annotation. Technical Report MS-CIS-99-01, Philadelphia, Pennsylvania, 1999.

[3] D. Ferrucci and A. Lally. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 2004.

[4] T. Götz and O. Suhre. Design and implementation of the uima common analysis system. *IBM Syst. J.*, 43(3):476–489, 2004.

[5] Ralph Grishman. Tipster text architecture design version 3.1, 1998.