UNIVERSIDAD DE LA REPÚBLICA

Facultad de Ingeniería

InCo - Instituto de Computación

"Sistema de Identificación de Segmentos Relacionados Temáticamente"

Tesis de grado presentada por

Martín Barreto - Ricardo Bedat

Docentes

Msc. Ing. Juan José Prada

Ing. Aiala Rosá

Montevideo, Uruguay - 2009

Resumen

El crecimiento de la cantidad de información digital se ha producido de manera exponencial y desordenada. Esto conlleva a una necesidad de herramientas y mecanismos que nos faciliten la tarea de recuperar y extraer información útil de manera eficaz y eficiente.

Un proceso eficaz para poder encontrar información útil es emplear mecanismos de Recuperación de Información seguido de procesos de Extracción de Información.

Nuestro trabajo se ubica dentro del área de Extracción de Información y tiene como objetivo principal la búsqueda de segmentos de texto que estén relacionados temáticamente a una expresión de consulta.

Desde un principio y durante todo el desarrollo del proyecto se optó por la construcción de una herramienta genérica y por tal motivo se consideró un dominio de documentos no acotado.

Durante el desarrollo del proyecto nos enfrentamos a desafíos de diversa índole vinculados al procesamiento del lenguaje natural; como son el reconocimiento de oraciones, la identificación de entidades con nombre, el reconocimiento de locuciones, el análisis de categoría gramatical, el reconocimiento de los significados de las palabras, la identificación del lema de una palabra, el reconocimiento de sinónimos, hiperónimos, merónimos y otros tipos de relaciones, la expansión de la consulta, descarte de palabras que no aportan información semántica, la desambiguación automática, entre otros.

Herramientas como *Freeling*[1] y *WordNet*[2] fueron claves para el éxito. Freeling nos brinda varios servicios para el análisis lingüístico y la posibilidad de poder mapear las palabras con conjuntos de sinónimos presentes en WordNet. WordNet nos permite encontrar las relaciones semánticas y léxicas entre conceptos del documento y de la expresión ingresada por el usuario.

También fue determinante para obtener una arquitectura flexible, robusta y escalable la utilización de *UIMA*[5]. Este framework es de gran ayuda para poder manipular información no estructurada y asignar metadatos al documento a medida que se procesa, así como también, para dividir el sistema en componentes independientes y reutilizables.

La utilización de UIMA nos permitió integrar nuestro sistema a *Lavinia*[7], que es un ambiente web basado en este framework para procesamiento del lenguaje natural desarrollado en el ámbito del Grupo de Procesamiento de Lenguaje Natural de la Facultad de Ingeniería[17].

Los resultados alcanzados por nuestro sistema son altamente alentadores, obteniendo una *precision* de 74,80 % y un *recall* de 78,28 %, valores superiores a trabajos relacionados[22, 23, 25, 24] realizados para otras lenguas.

Quedamos muy conformes con los resultados del trabajo, consideramos que tanto la investigación realizada como el producto obtenido son un valioso aporte al área, especialmente para el idioma español. Y aunque queda mucho por hacer y mejorar, nuestro proyecto es un buen punto de partida.

Palabras clave: *Identificación temática, Expansión de consultas, Extracción de Información, Recuperación de Información, Relación Semántica.*

Índice general

1 Introducción

1.1. El problema

El vertiginoso aumento en la cantidad de información no estructurada hace necesario el desarrollo de herramientas que faciliten extraer la información buscada, de forma rápida, precisa y efectiva. La lengua española es un lenguaje inmensamente rico que permite expresar una idea de múltiples formas. Esta gran riqueza es considerada como un gran obstáculo para el desarrollo de sistemas de extracción de información y sistemas de identificación temática, debido a la dificultad de que un software pueda inferir de forma automática el significado de los textos. En este sentido la extracción de información para el idioma español corre en desventaja frente a otros idiomas como el inglés, dado que la mayoría de los trabajos realizados están desarrollados y enfocados para este último. Debido a esto consideramos de gran importancia la investigación y desarrollo de técnicas de extracción de información para el idioma español.

El objetivo de este trabajo es lograr una herramienta capaz de extraer información de documentos en español que esté relacionada con una expresión de consulta. La idea principal consiste en poder marcar secciones en un texto que refieran al mismo tema que la expresión. La principal utilidad de este tipo de sistema es brindar apoyo al momento de encontrar información no estructurada en documentos de forma rápida, eficaz y eficiente.

Al ser nuestro trabajo un tipo particular de sistema de extracción de información, presenta con éstos un conjunto de problemas en común, como por ejemplo, el reconocimiento de entidades con nombre, el rol y las relaciones, la resolución de anáforas, el análisis de correferencias, etc. En la solución de los problemas antes mencionados, surge la necesidad de resolver otras dificultades de una granularidad menor, como pueden ser la segmentación del documento en palabras (tokenización), el análisis morfológico y la determinación del significado de las mismas, la identificación de las oraciones, entre otras.

1.2. Motivación

Aunque en los últimos años se ha investigado mucho en el área de la recuperación y la extracción de información, estos trabajos no están en su mayoría orientados al idioma español, lo que incentiva fuertemente a trabajar para esta lengua. También es importante mencionar que no hemos podido encontrar sistemas capaces de solucionar el problema abordado en este proyecto y consideramos que alcanzarlo es de gran valor en el desarrollo del área.

Por otro lado existen herramientas capaces de solucionar ciertos aspectos comunes en estos sistemas. Estas pueden brindar servicios lingüísticos básicos tales como reconocimiento de entidades con nombre, análisis morfológico y sintáctico, o mecanismos que permiten tener conocimiento semántico de un dominio concreto como ontologías y diccionarios.

Con respecto al desarrollo e implementación de sistemas de extracción de información, no está dicha la última palabra. Por el contrario, la metodología y las técnicas utilizadas son ambiguas, poco específicas y en muchos casos inexistentes. Sin embargo, la necesidad derivada del crecimiento exponencial en la cantidad de información existente, hace imprescindible la necesidad de avanzar en el área y desarrollar nuevos métodos y funcionalidades que permitan enfrentarse a esta situación. Esto nos motiva a incursionar en el área, aprender más de estos problemas y en lo posible colaborar en el desarrollo de soluciones.

1.3. Objetivo

El objetivo final de este trabajo es lograr a partir de una investigación, una herramienta capaz de extraer información de documentos en español que esté relacionada temáticamente con una expresión de consulta. Como consecuencia de lo anterior, estudiar y analizar el estado del arte en el área y realizar un análisis de las herramientas disponibles que faciliten el desarrollo de estos sistemas, ocupa un lugar importante en el trabajo.

Se espera que el producto construido sea capaz de identificar porciones de un texto que estén relacionadas con una expresión de interés para el usuario. Se pretende además que éste pueda integrarse a Lavinia[7] y de esta forma brindar sus funcionalidades en un ambiente web, así como poder ser utilizado por sistemas de mayor envergadura o futuros proyectos.

1.4. Plan de trabajo

En esta sección presentamos las diferentes etapas que transcurrieron en el desarrollo del proyecto. Esta será presentada por medio de un diagrama de Gantt para facilitar su entendimiento y visualización.

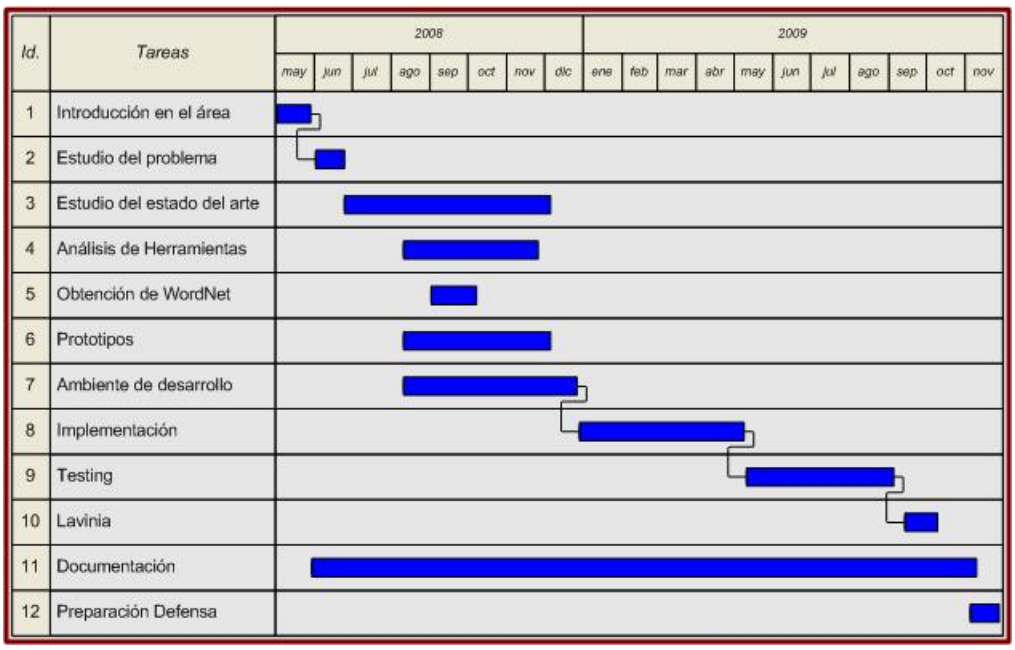


Figura 1.1: Evolución del proyecto

1. *Introducción al área.* La primera actividad realizada en el proyecto fue la de introducirnos en el área de investigación que propone el proyecto. Se estudiaron las nomenclaturas utilizadas en el área tanto para el idioma español, como para el inglés.

2. *Estudio del problema.* Una vez introducidos en el área realizamos el análisis completo del problema, tratando de especificar los requerimientos propuestos.

3. *Estudio del estado del arte.* Después de tener claro el propósito del proyecto se procedió a la búsqueda y estudio de trabajos relacionados, herramientas comúnmente utilizadas en proyectos similares, métodos y técnicas existentes.

4. *Análisis de herramientas.* Durante el desarrollo del estudio del estado del arte también se realizó un estudio en profundidad de herramientas utilizadas en proyectos similares y otras útiles para el desarrollo del proyecto.

5. *Obtención de WordNet.* En esta etapa gracias a la ayuda de los docentes (tutores del proyecto) se pudo obtener la licencia para el uso de Spanish WordNet[4].

6. *Prototipos.* Mientras se realizó el análisis de las herramientas se fueron construyendo prototipos, tratando de mitigar riesgos y permitir el desarrollo de la arquitectura. Algunos de los prototipos realizados fueron:

   - Prototipo de invocación a Freeling[1] a través de la compilación para Windows.
   - Prototipo de invocación a Freeling mediante la invocación al ejecutable.

- Prototipo de creación y utilización de la base de datos léxica Spanish WordNet.
- Prototipo de utilización e integración de UIMA[5].
- Prototipo de integración UIMA-Swing. Swing[8] se utiliza como tecnología para crear una ventana de desambiguación.
- Prototipo de integración de componentes UIMA con Lavinia.

7. *Ambiente de desarrollo.* Otra tarea realizada durante el análisis de herramientas y sobre todo durante el desarrollo de los prototipos fue la creación del ambiente de desarrollo que a posterior fue utilizado en la implementación del sistema.

8. *Implementación.* En esta etapa se diseñó e implementó el sistema. Se utilizaron todos los conocimientos obtenidos en las etapas previas y los prototipos antes desarrollados para lograr una correcta implementación de la solución del trabajo.

9. *Pruebas del sistema.* Una vez terminada la implementación se realizaron mediciones para analizar la *precision* del sistema y poder compararlo con otras soluciones.

10. *Lavinia.* Culminadas las pruebas del sistema, se trabajó en la incorporación de los componentes del sistema a Lavinia, permitiendo su utilización mediante una interfaz web.

11. *Documentación.* La documentación del proyecto se realizó durante todo el transcurso del mismo, pero puede separarse en tres partes bien definidas. La primera transcurre antes de la implementación y se enfocó en la creación de múltiples documentos independientes utilizados en el estudio del estado del arte y en el análisis de las herramientas. La segunda corresponde al período en que se realizó la implementación. En éste se realizaron documentos enfocados especialmente en la arquitectura y diseño del sistema. En la tercera y última parte se completó la documentación (agregando las pruebas y resultados), se ordenó y agrupó la documentación antes generada y se crearon los documentos finales.

1.5. Organización del informe

Este informe se compone de siete capítulos. Luego de este capítulo de *Introducción* se continuará con un detallado *Análisis del problema* a resolver mostrando las principales dificultades.

En el capítulo 3, correspondiente al *Marco teórico*, se presentan los principales conceptos de interés, enfocándonos en la recuperación y extracción de información. Se analizan trabajos relacionados, principalmente en el área de identificación de equivalencias semánticas entre textos e identificación temática de un documento, así como también problemas que surgen como subproblemas de los anteriores.

Basados en la información recopilada en el *Marco teórico,* el capítulo 4 *Solución propuesta*, presenta una posible solución teórica al problema.

El capítulo 5*, Desarrollo del proyecto,* se enfoca principalmente en la implementación de la solución propuesta, explica todas las herramientas, y dificultades de implementación que surgieron durante el desarrollo del proyecto, así como también justifica las distintas decisiones tomadas en algunos puntos clave del mismo.

En el capítulo 6, *Pruebas realizadas,* se presenta cómo se ponderaron las distintas relaciones, el conjunto de documentos y frases de consulta utilizados, resultados y estadíticas en función de fórmulas presentadas y algunas conclusiones derivadas de las pruebas.

Por último, el capítulo 7 *Conclusiones* presenta las conclusiones del proyecto así como también se plantean futuras mejoras y trabajos relacionados. Se describen los principales logros y dificultades encontradas.

2 Análisis del problema

> *"We can only see a short distance ahead, but we can see plenty there that needs to be done." - Alan Turing*

En este capítulo se presentará el problema y se realizará un análisis del mismo. Esto implica identificar las dificultades que deberán enfrentarse para lograr una solución completa.

Se propone realizar una aplicación que, dada una expresión de consulta y un texto, identifique los segmentos del texto que refieran al mismo tema que dicha expresión.

Por ejemplo se pretende que para la expresión *"aumento de precios"*, el sistema sea capaz de identificar fragmentos de texto como los siguientes:

"... reajustar el precio ..."
"... los costos habían aumentado ..."
"... aumento exorbitante de los precios ..."
"... se aumentarán los precios ..."
"... llevó al precio de muchos alimentos a incrementarse ..."
"... la suba de precios ..."
"... los incrementos de precios ..."
"... aumentos en los productos básicos ..."

Al analizar cuidadosamente el ejemplo presentado se puede reconocer una variedad de problemas, los cuales se dan a distintos niveles; desde el reconocimiento de palabras hasta el análisis de oraciones, y desde el entendimiento a nivel de oración hasta el análisis del texto completo. También podemos notar una dependencia entre estos niveles, es decir, para poder obtener un entendimiento global o a mayor escala del documento es necesario un entendimiento de sus componentes más básicos. Para poder llegar a inferir información a nivel de oraciones se necesita contar con información a nivel de las palabras o locuciones; para obtener información de los párrafos se deberá contar con información de palabras y oraciones del mismo.

Lo anterior nos lleva a un procesamiento *Bottom-Up* del documento. Cabe mencionar que menor escala no necesariamente significa menor complejidad. A medida que se vaya avanzando en la lectura de este documento se podrán comprender las dificultades inherentes al problema.

Una de las primeras dificultades encontradas es la separación del texto en palabras y la identificación de oraciones. Esto es necesario debido a que para poder contar

con la información global del documento se requiere información de los elementos que forman parte del mismo y los elementos de menor granularidad presentes son las propias palabras del texto.

Los principales problemas presentes se dan a nivel lingüístico y es en éstos que radican las principales dificultades de este proyecto. A continuación se presentan ejemplos de algunas de las dificultades lingüísticas.

Frase: *"La vida en ultramar es riesgosa."*

Texto: *"Los marineros se enfrentan a múltiples riesgos en su trabajo diario."*

En el primer ejemplo presentamos la dificultad de poder asociar las palabras *"ultramar"* con *"marinero"*. Si bien ambas palabras tienen significados diferentes, ellas comparten una idea en común. Palabras como estas y sumadas a otras como *"marejada"*, *"bajamar"* y *"marino"* forman parte de una misma familia léxica. Esto se debe a que todas estas palabras comparten una raíz léxica en común, *"mar"*. El reconocimiento de palabras de la frase de entrada y del texto que forman parte de una misma familia léxica nos permite identificar en el texto segmentos como el presentado. Sin embargo, la utilización de la raíz de una palabra a la hora de realizar la identificación no es suficiente como veremos en el siguiente ejemplo.

Frase: *"Ayer fueron al cine"*

Texto: *"Con Pedro íbamos al cine ayer pero al final vamos hoy."*

Se puede considerar que el texto del ejemplo responde satisfactoriamente la frase de consulta, sin embargo, la forma para relacionar las palabras *"fueron"* con las palabras *"íbamos"* y *"vamos"* no es trivial. Aunque estas palabras presentan la misma idea, éstas no tienen una raíz en común, pero sí comparten el mismo lema[1]: *"ir"*. La utilización del lema de una palabra y no la palabra en sí al momento de realizar la identificación de la frase en el texto permite la identificación de múltiples segmentos que no podrían ser identificados de otra manera.

Frase: *"Los monos están hambrientos."*

[1] Abstracción, a partir del haz de rasgos flexivos de una palabra, que representa a ésta como forma canónica. El lema de una palabra es usado como entrada de un diccionario o enciclopedia. El lema de un verbo es su infinitivo y el de un sustantivo el masculino singular. Por más información ver el Anexo A.3.

Texto: *"Les di plátanos a los monos, porque ellos estaban hambrientos."*

En el texto anterior ¿A quién se refiere la palabra *"ellos"*? Este problema tiene por nombre resolución de anáforas y se encuentra bajo el área de análisis de correferencias, que pretende encontrar las palabras o secuencias de palabras que hacen referencia a una entidad (objeto) anteriormente mencionado. Éste es un problema complejo. En este ejemplo la palabra "*ellos*" corresponde a los *"monos"* por lo que sería correcto identificar la frase de consulta en el texto. Siguiendo con el ejemplo.

Texto: *"Les di plátanos a los monos, porque ellos estaban pudriéndose"*

Como se puede ver, se cambió la palabra *"hambrientos"* por *"pudriéndose"*. Esto ocasionó casi involuntariamente que el *"ellos"* ya no hace más referencia a los *"monos"* sino a los *"plátanos"*. Esto muestra claramente las dificultades presentes en el análisis de correferencias, ya que no sólo depende de la estructura de la oración, sino también del contexto semántico de las palabras. Un sistema de identificación que no realice análisis de correferencias puede dejar de identificar una gran cantidad de segmentos donde aparezcan las entidades referenciadas por anáforas que en su mayoría se dan mediante pronombres.

Otro problema muy complejo y de suma importancia es poder identificar o inferir el significado de una palabra dentro de un contexto particular. Este problema se llama *desambiguación del significado.* La mayoría de estos problemas son generados por palabras polisémicas[2]. Ejemplos de esta problemática podrían ser:

"Esteban estaba en el banco por la tarde."

¿Cuál es el significado de *"banco"* (banco para sentarse, una entidad financiera, otro)? La respuesta a esta pregunta no se conoce debido a que el contexto es muy pobre, lo que muestra lo dificultoso del análisis de segmentos cortos como suelen ser las frases de consulta.

"Esteban estaba en el banco por la tarde, fue a depositar un cheque."

Siguiendo con el ejemplo, la nueva información del contexto y la situación planteada indican al lector que se trata de una entidad financiera. Pero esto sólo puede inferirse teniendo un conocimiento de la realidad presente en el texto, que indica que el depósito de un cheque se realiza en un banco.

Los problemas de desambiguación son muy importantes al momento de realizar la identificación de la frase de consulta del usuario en el texto. Una desambiguación incorrecta puede generar múltiples falsos positivos. Siguiendo con los ejemplos:

[2]Palabras polisémicas son aquellas que tienen múltiples significados. En el español la mayoría de las palabras son polisémicas por lo que para ser interpretadas correctamente se necesita información del contexto. Un ejemplo de palabra polisémica es *"pico"* que puede referirse al pico de un ave o al pico de una montaña entre otras acepciones posibles.

Frase: *"Esteban estaba en el banco."*

Texto: *"Esteban estaba en el banco por la tarde, fue a depositar un cheque. Esteban estaba sentado en una silla roja."*

Si consideramos la acepción de *"banco"* correspondiente a lugar para sentarse, sería correcto la identificación de la segunda oración del texto. En cambio, si consideramos la acepción correspondiente a entidad financiera se esperaría la identificación de la primera.

Otras problemáticas existentes surgen de la utilización en los documentos de sinónimos, antónimos, hipónimos, hiperónimos y otras relaciones semánticas[3] que generan complicaciones para determinar qué tan relacionada está la frase de entrada con oraciones o porciones del documento.

Frase: *"Se logró la meta"*

Texto: *"No se alcanzó el objetivo deseado"*

Para la frase y texto presentado, ¿debería el sistema identificar la frase en el texto? En primera instancia tenemos el problema de que las palabras clave de la frase *"logró"* y *"meta"* no aparecen en el texto, pero sí aparecen sinónimos de ellas. Esto trae la necesidad de poder identificar no solamente las palabras en el documento sino también poder identificar sus sinónimos. Pero aunque se logre identificar la palabra con el apoyo de los sinónimos, el texto comienza con una negación ¿Habría que descartar la identificación a causa de la negación? ¿Qué ocurre si hay múltiples negaciones? ¿Y si ocurren antónimos negados? Esto es todo un dilema. Por un lado la negación del opuesto de una palabra no tiene por qué coincidir con el significado de la palabra. Por otro lado estos casos tienen una estrecha relación semántica y podrían ser sumamente interesantes para el usuario que realizó la consulta.

Otro obstáculo similar ocurre con el uso de los hiperónimos e hipónimos, por ejemplo

Frase: *"Tengo un automóvil rojo."*

Texto: *"Tengo un vehículo de dos ruedas rojo y un descapotable blanco"*

¿Para este ejemplo sería correcto identificar en el texto la frase de la consulta? En principio el texto no contiene ninguna de las palabras clave de la frase de consulta, pero sí contiene un hiperónimo *"vehículo"* y un hipónimo *"descapotable"* correspondientes a *"automóvil"*, o sea que para poder realizar la identificación partimos de la

[3]Estas relaciones son explicadas en la sección 3.1, en el Anexo A.3 y en el Glosario.

necesidad de identificar los hiperónimos e hipónimos de las palabras de una consulta. Esto implica contar con algún mecanismo que permita identificar éste y otros tipos de relaciones. Notoriamente, este recurso tendría una gran influencia en los resultados de nuestro sistema. A parte de esto, aunque en el texto aparece la palabra *"rojo",* ésta no aparece asociada explícitamente a un automóvil, sino que a un vehículo. Situaciones como ésta pueden generar falsos positivos. Relaciones como hiperónimos e hipónimos son muy comunes y como éstas, existen muchas otras, que generan un sin número de dificultades similares.

Frase: *"Gato negro."*

Texto: *"Ayer vi un felino, éste tenía las patas de color oscuro."*

En la frase de consulta brindada por el usuario del sistema se tienen casi los mismos inconvenientes que están presentes en los textos. ¿Qué significa *"gato"* en ese contexto? ¿A qué entidad referencia la palabra *"éste"*? Con relación al texto, sería deseable poder inferir que *"gato"* es hipónimo de *"felino",* que los *"gatos"* poseen *"patas"* (relación parte todo) y que negro es un color oscuro.

Frase: *"Mujeres lindas de ojos claros."*

Texto: *"Fernanda es una rosa y tiene los ojos como el cielo."*

Otros fenómenos que dificultan la inferencia de relaciones son el uso de metáforas y las ocurrencias de metonimias[4]. En el ejemplo presentado se expresa que los ojos de Fernanda son como el cielo, lo que implica que estos son de un color celeste claro. La ocurrencia de metáforas en el texto ocasionan que parte de la información del texto sea ambigua y no esté presente explícitamente en el texto. Poder realizar este tipo de identificación para una persona es posible gracias al conocimiento del mundo que posee. Pero lograr que un sistema automático sea capaz de resolverlo de forma satisfactoria es sumamente complejo y todavía está lejano.

Por último, podemos mencionar otros problemas que hay que tener presente y ser conscientes de su existencia antes de comenzar con el análisis automático de textos. Algunos de ellos son: formato de texto, errores de ortografía, capitalización (uso de

[4]La metonimia o "trasnominación" es un fenómeno de cambio semántico por el cual se designa una cosa o idea con el nombre de otra, sirviéndose de alguna relación semántica existente entre ambas. Es una figura retórica que consiste en designar una cosa con el nombre de otra con la que mantiene una relación de proximidad o contexto: si decimos *"hay que respetar las canas"* por *"hay que respetar la vejez"*. La diferencia con las metáforas es que estas usan la semejanza como relación semántica en cambio la metonimia puede hacer uso de relaciones causa-efecto, parte-todo, autor-obra, continente-contenido.

mayúsculas y minúsculas), símbolos, palabras en otros idiomas, aparición de locuciones, análisis de categoría gramatical, identificación de entidades con nombre, determinación de la forma raíz y el lema de una palabra, realizar análisis de características léxicas y estructura semánticas, entre otros.

Todas las dificultades presentadas tienen como objetivo comprender el problema que implica la realización de un análisis lingüístico automático y nos sirve como punto de partida para el desarrollo del proyecto.

En el capítulo siguiente analizaremos trabajos relacionados que se enfrentan a algunas dificultades aquí presentadas.

3 Marco teórico

"Know where to find the information and how to use it - That's the secret of success" - Albert Einstein

La extracción de información (EI) es una disciplina dentro del procesamiento del lenguaje natural (PLN) que supone una revolución tecnológica en el ámbito de la recuperación de información (RI) y que pretende agilizar la obtención de la información útil por parte de los usuarios [34]. Tradicionalmente, los usuarios recuperan una gran cantidad de información y después, manualmente, deben extraer la información de estos documentos tras el análisis de los resultados recuperados. Aplicando la tecnología existente de extracción de información automática se pretende filtrar automáticamente los resultados tornando la labor descrita anteriormente menos costosa. Los sistemas de EI realizan las tareas de buscar información concreta en colecciones de documentos, detectar la información relevante, extraerla y presentarla en un formato susceptible de ser tratado automáticamente posteriormente. Si bien los sistemas de EI pueden extraer información errada o no identificar ciertos eventos o relaciones que son de interés, igualmente tienen un gran valor. Pueden ser utilizados para extraer información para luego ser verificada por un humano y de esta forma ahorrar una gran cantidad de tiempo. Los beneficios de un sistema automatizado, aunque sus soluciones no sean del todo correctas comparados con la extracción puramente manual, ya son argumentos bastante fuertes para su investigación y desarrollo.

Se puede considerar que el desarrollo de sistemas de EI para el idioma español está recién en sus principios, ya que existen muy pocos trabajos relacionados al respecto, la información existente es muy escasa y existen pocas herramientas útiles. Esto trae aparejado un conjunto de dificultades que deberán salvarse para lograr los objetivos de este trabajo, pero también nos incentiva a seguir involucrandonos ya que, consideramos que los logros alcanzados van a contribuir con el desarrollo del área, así como también a fomentar futuras investigaciones.

3.1. Conceptos básicos

Para comprender el marco teórico que comprende el procesamiento del lenguaje natural y especialmente los sistemas de recuperación y extracción de información es necesario el conocimiento de ciertos conceptos básicos que se usan con frecuencia en estas áreas. En esta sección presentaremos los más importantes.

Un lector que ya posea conocimientos sobre estos temas podrá omitir esta sección. La información de los conceptos presentados fue recopilada de diversas fuentes como: Speech and Languaje Processing[21], Wikipedia[13], Real Academia Española[16], Encarta[12], WordReference[14], WikiLengua[15], entre otras.

Procesamiento del Lenguaje Natural

Es una subdisciplina de la inteligencia artificial y la rama ingenieril de la lingüística computacional. Se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas o entre personas y máquinas por medio de lenguajes naturales.

Busca crear programas que puedan analizar, entender y generar lenguajes que los humanos utilizan habitualmente, de manera que el usuario pueda llegar a comunicarse con el ordenador de la misma forma que lo hace con un ser humano.

Las posibles aplicaciones del PLN son muy variadas, ya que su alcance es muy grande. Algunos ejemplos de las aplicaciones son: traducción automática, recuperación de la información, extracción de información, resúmenes, reconocimiento de voz, etc.

Recuperación de Información

La recuperación de información (RI) es la búsqueda manual o automática de información en documentos. Los sistemas automáticos de RI son usados para reducir la sobreinformación (*information overload*). Muchas universidades y bibliotecas públicas usan sistemas de RI para proveer un acceso a libros, diarios y otros documentos. Las herramientas de búsqueda en la web como Google y Yahoo son algunas de las aplicaciones de RI más utilizadas. Los procesos de RI comienzan cuando un usuario ingresa una consulta en el sistema. Hay que tener en cuenta que en RI una consulta no identifica a un único documento, sino que múltiples documentos son devueltos como resultado del proceso RI que satisfacen con mayor o menor relevancia la consulta realizada.

Extracción de Información

La EI tiene como objetivo encontrar, relacionar y extraer información relevante de forma automática y presentarla en un formato adecuado para ser procesada posteriormente. Por ejemplo información que se encuentre categorizada, contextualizada y semánticamente bien definida, de un dominio específico, a partir de documentos no estructurados. La importancia de la EI está determinada por el aumento de información no estructurada disponible, principalmente en Internet. Toda esta información puede ser fácilmente accesible estructurándola a un modelo relacional o representándola mediante un documento XML. Una aplicación típica de estos sistemas es analizar un conjunto de documentos escritos en lenguaje natural y automáticamente poblar una base de datos con la información extraída.

Análisis morfológico

Es una clase de análisis gramatical que consiste en determinar las posibles clases o categorías gramaticales de cada palabra de una oración.

El procedimiento básico para realizar un análisis morfológico es el siguiente:

- Se obtiene cada palabra de cada oración.
- Se determina a qué clases pertenece cada palabra: sustantivo, adjetivo, pronombre, verbo, adverbio, preposición, etc.
- Si se trata de una palabra variable, se analizan sus accidentes gramaticales: el género, el número, la persona, el tiempo, el modo, etc.

Las clases de palabras variables (con flexión) son: sustantivos, artículos, adjetivos, pronombres, verbos. Las clases de palabras invariables son: adverbios, conjunciones, preposiciones, interjecciones.

Categoría Gramatical

Categoría gramatical (*POS, Part of Speach*) es una clasificación morfológica de las palabras según su tipo. La gramática tradicional distingue nueve categorías gramaticales:

- Determinante (el, la, un, unos, esos, nuestro, algún)
- Preposición (a, de, por, contra)
- Pronombre (yo, él, mí, se, aquello, donde, que)
- Conjunción (y, o, si, que)
- Nombre (casa, casas, felicidad)
- Verbo (cantar, cantando, cantaremos, sabe, sé)
- Adjetivo (alto, alta, lindos, inteligente, solo)
- Adverbio (medio, simplemente, sólo)

POS tagging

Part-of-speech tagging, también llamado etiquetado gramatical, es el proceso de asignar o etiquetar a cada una de las palabras de un texto su categoría gramatical. Este proceso se puede realizar en base a la definición de la palabra o el contexto en que aparece, por ejemplo su relación con las palabras adyacentes en una frase, oración o párrafo. Uno de los usos de este etiquetado tiene lugar en el contexto de la lingüística computacional, mediante el empleo de algoritmos que realizan el etiquetado en base a etiquetas descriptivas predefinidas.

El proceso de etiquetado gramatical es más complejo de lo que parece a primera vista, puesto que no se reduce a tener una lista de palabras y sus correspondientes categorías gramaticales, debido a que algunas palabras pueden tener distintas categorías gramaticales en función del contexto en que aparecen. Este hecho suele ocurrir a menudo en el lenguaje natural, donde una gran cantidad de las palabras son ambiguas. Por ejemplo, la palabra *"dado"* puede ser un nombre singular o una forma del verbo *"dar"*.

A diferencia del análisis morfológico, el *POS tagging* determina una única categoría gramatical para cada palabra, utilizando información del contexto para su elección.

Red semántica

Una red semántica o esquema de representación en red es una forma de representación de conocimiento lingüístico en que a las interrelaciones entre diversos conceptos o elementos semánticos se les da la forma de un grafo. Estas redes pueden ser visualizadas como grafos, aunque algunas veces pueden ser también árboles. Las redes semánticas pueden ser mapas conceptuales y mentales. En un grafo o red semántica los elementos semánticos se representan por nodos. Dos elementos semánticos entre los que se admite la relación semántica que representa la red, estarán unidos mediante una línea, enlace o arista. Cierto tipo de relaciones no simétricas requieren grafos dirigidos que usan flechas en lugar de líneas.

Taxonomía

Una taxonomía es un tipo de vocabulario controlado en que todos los términos están conectados mediante algún modelo estructural (jerárquico, arbóreo, etc.) y especialmente orientados a los sistemas de navegación, organización y búsqueda de contenidos en los sitios web.

La construcción de una taxonomía supone la realización de cuatro procesos:

- Delimitación de la realidad.
- Extracción del conjunto de términos o categorías que representan dicha realidad.
- Control terminológico de los términos o categorías.
- Establecimiento del esquema y la estructura del domino de acuerdo a los términos y categorías.

El objetivo principal de la creación de una taxonomía es representar la realidad o dominio. Ésta juega un papel protagonista en la búsqueda de información.

Tesauro

Un tesauro (del latín *thesaurus* que significa tesoro) es una lista que contiene los términos empleados para representar los conceptos, temas o contenidos de los documentos, con miras a efectuar una normalización terminológica que permita mejorar

el canal de acceso y comunicación entre los usuarios y las unidades de información (biblioteca, archivo o centros de documentación). Es un intermediario entre el lenguaje natural y el que emplean los especialistas de un determinado campo del saber (lenguaje controlado).

Aunque en la práctica tradicional se habla de unitérminos, en la actualidad se han efectuado grandes variaciones dando incorporación a términos o descriptores compuestos, es decir, descriptores que se componen de dos o más palabras. Los términos que conforman el tesauro se interrelacionan entre ellos bajo tres modalidades de relación:

- Relaciones jerárquicas: establece subdivisiones que generalmente reflejan estructuras de parte-todo.
- Relaciones de equivalencia: determina relaciones de sinonimia, homonimia, antonimia y polinimia entre los términos.
- Relaciones asociativas: mejoran las estrategias de recuperación y ayudan a reducir la polijerarquía entre los términos.

En líneas generales, un tesauro comprende lo siguiente:

- Un listado de términos preferidos, que se los ordena en forma alfabética, temática y jerárquica.
- Un listado de sinónimos de esos términos preferidos llamados descriptores.
- Una jerarquía o relaciones entre los términos.
- Definiciones de los términos, para facilitar la selección de los mismos por parte del usuario.
- Un conjunto de reglas para usar el tesauro.

Las entradas de un tesauro no deben ser consideradas sólo como una lista de sinónimos. Son obras en las que se relacionan numerosas palabras que guardan una relación más o menos directa con la palabra objeto de consulta. No son diccionarios de sinónimos, ya que estos últimos incluyen únicamente palabras con un significado similar y equivalente.

Ontología

El término ontología, en informática, hace referencia a la formulación de un exhaustivo y riguroso esquema conceptual dentro de un dominio dado, con la finalidad de facilitar la comunicación de la información entre diferentes sistemas. Las ontologías catalogan y definen los elementos y entidades que existen en un cierto dominio, así como sus relaciones y propiedades. Existen ontologías específicas (de términos

médicos, empresariales, aeronáuticos, etc.) y ontologías de carácter general. Las ontologías favorecen la comunicación entre personas, organizaciones y aplicaciones porque proporcionan una comprensión común de un dominio, de modo que se eliminan confusiones conceptuales y terminológicas.

Una definición de ontología que se ha convertido en estándar es la presentada por Gruber[31] en 1993, donde una ontología constituye *"a formal, explicit specification of a shared conceptualization"*.

Una definición más concreta fue realizada por Weigand[35] en 1997: *"An ontology is a database describing the concepts in the world or some domain, some of their properties and how the concepts relate to each other"*.

En 1998 Steve[36] distingue tres tipos fundamentales de ontologías:

- Ontologías de un dominio, en las que se representa el conocimiento especializado pertinente de un dominio o subdominio, como la medicina o las aplicaciones militares.
- Ontologías genéricas, en las que se representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos.
- Ontologías representacionales, en las que se especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan meta-ontologías (*meta-level* o *top-level ontologies*).

A estos tres tipos, Guarino[37] añade las ontologías que han sido creadas para una actividad o tarea específica (denominadas *task ontologies*), como por ejemplo la venta de productos o el diagnóstico de una enfermedad y las ontologías creadas para una aplicación específica.

Diccionario

Un diccionario es un catálogo de consulta de términos que se encuentran generalmente ordenados alfabéticamente. De dicha compilación de palabras o términos se proporciona su significado, etimología, ortografía y, en el caso de ciertas lenguas, su pronunciación y separación silábica. La disciplina que se encarga, entre otras tareas, de elaborar diccionarios es la lexicografía. La información proporcionada por un diccionario varía según el tipo de diccionario del que se trate.

Relaciones léxicas

Sinonimia

La sinonimia es una relación semántica de identidad o semejanza de significados entre determinadas palabras (llamadas sinónimos) u oraciones. Por lo tanto sinónimos

son palabras que tienen un significado similar o idéntico entre sí, y pertenecen a la misma categoría gramatical.

Por ejemplo, "fútbol" y "balompié" son sinónimos.

Hiponimia

En semántica lingüística, se denomina hipónimo a aquella palabra que posee todos los rasgos semánticos de otra más general, su hiperónimo, pero que además añade en su definición otros rasgos semánticos que la diferencian de la segunda. Por ejemplo, *descapotable* es hipónimo de *coche*, ya que comparte todos sus rasgos mínimos (vehículo, con motor, pequeño tamaño, etc.), pero añade a estos el rasgo *"sin capota"*.

Un ejemplo: *"Lunes"*, *"Martes"*, *"Miércoles"* son hipónimos de *"día"*

Hiperonimia

Se denomina hiperónimo a aquel término general que puede ser utilizado para referirse a la realidad nombrada por un término más particular. Un hiperónimo no posee ningún rasgo semántico que no comparta su hipónimo, mientras que éste sí los posee. Por ejemplo, *coche* posee las *propiedades "vehículo"*, *"con motor"* y *"pequeño tamaño"*, que comparte con *descapotable*, mientras que, *descapotable* posee además el rasgo *"sin capota"*, que lo diferencia de *coche*. Al redactar un texto conviene utilizar hiperónimos para evitar la repetición de palabras ya empleadas anteriormente, como se hace en el siguiente ejemplo:

"De repente, un descapotable rojo paró frente al banco. Del automóvil salieron dos individuos encapuchados, mientras otro esperaba en el vehículo."

3.2. Sistemas de Extracción de información

A grandes rasgos nuestro trabajo se basa en la creación de un sistema de EI capaz de extraer información relacionada con una frase de consulta. Consideramos de suma importancia brindar una definición de EI y explicar sus componentes principales desde una perspectiva de la arquitectura y su interacción en un flujo.

Cuando se desea obtener información concreta en un conjunto de documentos, normalmente se aplican técnicas de RI, a partir de lo cual se obtienen los documentos de interés, y luego métodos de EI, con lo cual se obtiene información concreta en dichos documentos. Esta información extraída típicamente se refiere a entidades, relaciones y eventos de interés, y está restringida a un dominio acotado.

La EI tiene como objetivo obtener información útil o relevante de forma automática y presentarla en un formato adecuado para ser procesada posteriormente. Esta salida normalmente es estructurada, en la mayoría de los casos en formato XML.

Un sistema de EI se basa en la aplicación de un conjunto de reglas, estas pueden ser construidas tanto de forma manual (reglas estáticas) como automáticamente mediante algún mecanismo de aprendizaje automático. Estas reglas están basadas en aspectos del lenguaje, de la semántica y del estilo de escritura propios de cada dominio [38]. Por esto la utilización de técnicas para el procesamiento del lenguaje natural y el uso de conocimiento relacionado con el dominio en que se está trabajando son de vital importancia para su construcción.

Los sistemas de EI se pueden aplicar a documentos no estructurados (la mayoría de los casos) o estructurados. Los documentos no estructurados, es decir los que se encuentran en lenguaje natural, son el objetivo principal de los sistemas de EI. La EI para documentos estructurados es una tarea más sencilla dada la naturaleza estructurada del mismo, el cual contiene metadatos incluidos, lo que facilita la obtención de información. La EI en documentos no estructurados es más compleja.

La mayoría de los sistemas de EI deberán enfrentarse a los mismos problemas. Es por esto que éstos tradicionalmente se han separado en componentes o módulos, cada uno encargado de enfrentarse a una problemática específica. Es importante destacar que dependiendo del sistema y el problema general que enfrente puede suceder que no todos los módulos (que se presentarán a continuación) sean necesarios e incluso se pueden agregar otros con motivo de resolver problemas particulares.

Reconocimiento de entidades con nombre

El reconocimiento de entidades con nombre (*Named Entity Recognitio*n), es una subtarea de la EI. Se encarga de buscar y clasificar elementos atómicos, es decir, la información de nombres, lugares, organizaciones, expresiones temporales, expresiones numéricas, entre otras. Esta información es sumamente importante para poder resolver consultas directas del tipo ¿Quién? ¿Cuándo? ¿Cuánto? y ¿Dónde?. Una vez recuperada esta información se habrá obtenido una gran cantidad de datos sobre el documento procesado. Generalmente estos serán utilizados por otros sistemas más complejos que abordan la RI, como puede ser un sistema de Pregunta-Respuesta.

En la actualidad, la tarea del reconocimiento de entidades con nombre se presenta como bastante exacta. Tiene actualmente un promedio de exactitud del 90 % en la conferencia MUC-7[1]. Esta etapa en un sistema tratado en dicha conferencia contaba con un error de apenas un 6.61 %, es decir con una tasa de acierto de 93.39 %. Es importante destacar que estos resultados son para sistemas de EI basados en un dominio específico, es decir, a priori conocen el dominio del documento que se va a procesar. En caso de no contar con un dominio acotado, como es el caso de nuestro proyecto, este proceso se presume que no posee un nivel de exactitud tan elevado.

Esta etapa se implementa comúnmente mediante dos enfoques ligeramente distintos, una de las técnicas es utilizando gramáticas basadas en conocimiento lingüístico y la otra es utilizando modelos estadísticos. El primer enfoque presenta mejores prestaciones que el segundo.

[1] Ver 3.2.1.

Resolución de correferencias

Las correferencias son dos expresiones referentes que son usadas para referirse a la misma entidad.

"Les dimos bananas a los monos porque ellos estaban hambrientos."

En este ejemplo, las palabras *"monos"* y *"ellos"* son correferentes y la palabra *"ellos"* es una anáfora de la palabra *"monos"*.

La resolución de correferencias (*Coreference Recognition*) tiene como principal tarea identificar palabras correferentes. Esta etapa resuelve las anáforas, es decir, intenta determinar el significado de los pronombres. Este problema es muy complejo de resolver debido a que las anáforas son comunes en un texto y no es fácil determinar a qué elemento está haciendo referencia el pronombre sin comprender el contexto en el que está presente, e incluso aún, considerando el contexto, puede ser una tarea compleja.

Extracción de terminología

La extracción de terminología consiste en la identificación de las palabras clave o términos relevantes de un documento. Estos sistemas generalmente hacen uso del procesamiento lingüístico para obtener las palabras clave candidatas, las cuales son comparadas con un conjunto preestablecido de claves de un dominio. Es ampliamente utilizado debido a su poca ambigüedad y alta especificación. Aparece en múltiples sistemas como: similitud semántica, administración del conocimiento, traducción automática.

Para dominios no específicos, la tarea puede ser mucho más compleja. Existen trabajos que se enfrentan a ésta. Generalmente y a grandes rasgos, se basan en la utilización de árboles jerárquicos de relaciones de hiperónimos para cada una de las palabras del documento, intentando establecer qué palabras, en el conjunto de los árboles, aparecen con mayor frecuencia.

Reconocimiento de rol

El reconocimiento de rol tiene como objetivo identificar *quién* hizo *qué* dentro de un texto. Se propone identificar qué papel tiene una persona, objeto o lugar dentro del texto. Por ejemplo dado el texto *"Mario le compró el auto a Juan"* busca identificar quién es el comprador y quién es el vendedor.

Reconocimiento de relaciones

Busca encontrar relaciones semánticas entre entidades del texto.
Ejemplo de esto puede ser:
Para la relación "PERSONA **vive en** CIUDAD".
Y el texto:

"Mariana vive con su familia en Montevideo. Arturo es el padre de Mariana y es oriundo de Fraile Muerto.".

El sistema debería detectar las relaciones:

"Mariana **vive en** Montevideo" y "Arturo **vive en** Montevideo".

Un enfoque para resolver esta tarea es utilizar ontologías para un dominio determinado.

Las tareas antes mencionadas son las principales subtareas de un sistema de EI y cada una puede considerarse como un problema independiente y complejo.

3.2.1. MUC

Las conferencias MUC (*Message Understanding Conference*)[11] que se desarrollaron entre 1987 y 1998, tenían como objetivo evaluar el estado del arte de los sistemas de EI y fomentar el desarrollo de nuevos y mejores métodos de EI.

Los organizadores de dichas conferencias brindaban un dominio de aplicación para los sistemas de EI además de definir las reglas de las tareas de la extracción. De hecho, han creado un dominio de aplicación con un corpus de textos etiquetados con la información a extraer y un conjunto de textos para evaluar las aplicaciones de las organizaciones participantes. En esta conferencia se optó por la utilización de plantillas atributo-valor para la evaluación de los sistemas. De esta manera, se puede comprobar la exactitud del sistema, comparando las salidas del sistema con unas plantillas que han sido generadas manualmente.

Las tareas que se llevaban a cabo eran:

- *scenario template*: rellenar una plantilla con información relevante. Por ejemplo tipo de evento, agente, tiempo, etc.
- *named entity recognition*: extraer entidades como nombres de personas, organizaciones, fechas, etc.
- *template element*: rellenar una plantilla con los atributos de una entidad.
- *coreference resolution*: resolución de correferencias.
- *template relation*: extracción de relaciones entre entidades.

Cada instancia de la conferencia brindaba un dominio sobre el cual los sistemas de los participantes competían para ver cuál lograba los mejores resultados, según las especificaciones impuestas en la tarea de extracción de información. Los dominios presentados han sido:

- Textos sobre operaciones navales [MUC-1, 1987 y MUC-2, 1989]
- Noticias sobre actividades terroristas [MUC-3, 1991 y MUC-4, 1992]
- Noticias sobre microelectrónica y fusión de corporaciones [MUC-5, 1993],
- Artículos sobre sucesión de puestos en compañías importantes [MUC-6, 1995]
- Artículos sobre vehículos espaciales y lanzamiento de misiles [MUC-7, 1997]

3.2.2. TREC

TREC (*Text REtrieval Conference*)[9] es una conferencia de recuperación de textos. Está co-patrocinada por el NIST (*National Institute of Standards and Technology*)[10] y el Departamento de Defensa de los Estados Unidos. Se inició en 1992 como parte del programa de texto TIPSTER[2].

El propósito del TREC es la de proveer la infraestructura necesaria para la evaluación a gran escala de metodologías de RI. En particular se persiguen los siguientes objetivos:

- Animar la investigación de sistemas de RI basados en grandes colecciones de textos.
- Incrementar la comunicación entre la industria, el sector académico y el gobierno, al crear un foro abierto que permite el intercambio de investigaciones e ideas.
- Aumentar la transición de la tecnología desde los laboratorios de desarrollo hacia la construcción de productos comerciales, a través de la realización de demostraciones, presentando las mejoras substanciales de las metodologías de RI en problemas del mundo real.
- Aumentar la disponibilidad de técnicas apropiadas para la evaluación de estos sistemas desarrollados por la industria y el sector académico.

TREC es supervisado por un comité de programa integrado por representantes del gobierno, la industria y la academia. Para cada participante del programa, NIST proporciona un conjunto de documentos de prueba y preguntas que permitan la recuperación. Los participantes ejecutan el conjunto de textos de pruebas en sus propios sistemas de recuperación de los datos, y devuelven a NIST una lista con los mejores documentos clasificados que se encontraron en la ejecución. NIST revisa los resultados individuales, juzga la exactitud de los documentos recuperados, y evalúa los resultados. El ciclo TREC finaliza con un taller que es un foro para que los participantes compartan sus experiencias.

[2]http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/

3.2.3. ACE

La conferencia ACE (*Automaric Content Extraction evaluation*) fue realizada por primera vez en 1999 y también se dedica a evaluar sistemas de EI. Entre esos sistemas se destacan sistemas de reconocimiento de habla automática (*automatic speech recognition*) y reconocimiento óptico de caracteres (*optic character recognition*). Las principales tareas de EI realizadas en esta conferencia son: reconocimiento de entidades, traducción de entidades, reconocimiento de relaciones, extracción de eventos, y extracción de información a partir de sistemas de reconocimiento de habla automática y de reconocimiento óptico de caracteres.

3.2.4. DUC y TAC

Las conferencias DUC (*Document Understanding Conferences*) comenzaron en el año 2001 y se llevaron a cabo hasta el año 2007, desde el 2008 las tareas se incorporaron a la conferencia TAC (*Text Analysis Conference*). Evaluaban sistemas de resumen automático (*summarization*) a partir de un *topic*.

En el 2003 se incorporo la tarea de generar un resumen automático como respuesta a una pregunta y en el 2009 se incorpora la tarea de población de bases de conocimiento (*Knowledge Base Population*), que junta tareas de búsqueda de respuestas con tareas de EI.

3.3. Trabajos relacionados

En esta sección se presentarán los trabajos que están relacionados con las problemáticas del proyecto. La descripción completa con ejemplos puede encontrarse en el Anexo A, *Estudio del estado del arte*.

Aunque los trabajos de EI realizados en español son escasos, se han desarrollado muchos trabajos en otras lenguas, especialmente en inglés. Estos fueron de gran utilidad para el desarrollo de nuestro proyecto.

Para comenzar nos parece importante destacar la arquitectura para un sistema de EI genérico, presentada en el trabajo "*Técnicas y desafíos en sistemas de extracción de información*"[27] (por más información ver Anexo A.1.2). Ésta consiste en un flujo que se compone de las siguientes etapas:

- *Análisis léxico*. En esta etapa se divide el texto en oraciones y éstas en palabras. Luego a cada palabra se le determina la categoría gramatical y otras características. Este proceso puede hacer uso de diccionarios de propósito general y diccionarios especiales, tales como diccionarios de empresas, de lugares, de nombres propios comunes.

- *Reconocimiento de entidades con nombre*. En la segunda etapa se pretende identificar los nombres propios, fechas, y números. Los nombres aparecen frecuentemente en los documentos e identificarlos y clasificarlos son de las principales

tareas de la mayoría de los sistemas de EI. Los principales métodos utilizados para el reconocimiento de estas entidades es el uso de expresiones regulares, estructura sintáctica y características ortográficas como son el uso de mayúsculas o minúsculas.

- *Análisis de estructura sintáctic*a. La identificación de ciertos aspectos de la estructura sintáctica puede simplificar significativamente las etapas siguientes. Por otro lado, la identificación completa de la estructura sintáctica es una tarea dificultosa. Como resultado hay una gran variación en la cantidad de estructura que es específicamente identificada y algunos sistemas no poseen una etapa separada de análisis sintáctico. En sí el análisis de la estructura sintáctica es una clase de análisis gramatical, estudia la función de las palabras y los distintos grupos de palabras que componen la oración. Un clásico ejemplo sería la determinación del sujeto y predicado de una oración.

- *Escenario de mapeo de patrones (Scenario Pattern Matching)*. En esta fase se pretende encontrar relaciones y roles entre diferentes entidades con nombre. Busca extraer información al encontrar relaciones que *mapean* dos o más entidades en el texto analizado. Un ejemplo es *"Juan Pérez se retiró como ejecutivo de vice presidencia, en su reemplazo entró Manuel González*" un resultado esperado de este proceso es poder identificar que *Manuel González* es el nuevo ejecutivo de vice presidencia.

- *Análisis de correferencias*. Esta fase identifica frases que referencian a la misma entidad. Básicamente lo que resuelve son las anáforas dadas por pronombres y nombres.

- *Inferencia e integración de eventos*. En muchas situaciones, ocurre que la información parcial de un evento puede aparecer dispersa en las oraciones del documento. Esta información debe ser combinada antes de poder extraerla correctamente. En otras circunstancias la información sólo aparece de forma implícita y debe hacerse explícita a través del proceso de inferencia.

Si bien los módulos presentados anteriormente son propios de los sistemas de EI más formales, en la práctica muy pocos sistemas implementan todas las etapas descriptas ya que cada una presenta una gran complejidad. Sin embargo, la modulación descripta en este trabajo es muy utilizada en este tipo de sistemas, aunque no con los mismos módulos.

Dentro del área de la EI también podemos encontrar la problemática de lograr identificar el tema del que trata un texto. Existe una variedad de técnicas utilizadas que permiten resolver este problema. Estas técnicas pueden ser clasificadas en *técnicas independientes del dominio* y *técnicas dependientes del dominio*.

Dentro de las *técnicas independientes del dominio* se pueden encontrar: identificación temática por frecuencia de ocurrencia de palabras [25] o técnicas que hacen uso

del principio de localidad[28]. El principio de localidad hace uso de información recopilada tras el estudio de múltiples documentos. En este estudio se logra identificar en que sección de un documento existe mayor probabilidad de que se encuentren las palabras clave que identifique el tema del texto analizado.

Una característica relevante de la técnica que se basa en el conteo de palabras (frecuencia de palabras) es que no capta la generalización de los conceptos, es decir no tiene conocimiento de las relaciones (por ejemplo: hiperónimos, sinónimos) que se dan entre conceptos y términos del documento. Para poder tener mayor conocimiento se necesita un nivel alto de análisis gramatical y algún mecanismo para poder inferir relaciones entre los conceptos.

En el conjunto de *técnicas que hacen uso de un dominio específico*[3] se encuentran técnicas que hacen uso de jerarquías, ontologías y taxonomías [22, 23]. La idea principal de estas técnicas es la de mapear las correspondencias semánticas entre palabras del documento y los conceptos en el dominio.

A continuación haremos una breve presentación de trabajos relacionados que hacen uso de cada una de esas técnicas, enfatizando en los puntos a favor y en contra que poseen.

3.3.1. Presentación y análisis de trabajos relacionados

Una investigación que es interesante de analizar es la realizada por Sabrina Tium, Rosni Abdullah y Tang Eaya Kong de la Universidad Sains de Malasia, *Automatic Topic Identication Using Ontology Hierarchy* [22] que plantea un método para la identificación automática del tema de un documento web. En su proyecto utilizaron una ontología relacionada para poder identificar el tema del documento. Un aspecto a destacar es el modo en que enriquecieron la ontología por medio de la utilización de relaciones semánticas encontradas en WordNet.

Para identificar el tema principal de un documento a partir de la exploración de la estructura jerárquica de la ontología se encontraron con el inconveniente de poder mapear la representación del documento con los conceptos presentes en la ontología, es decir mapear las palabras clave del documento con la ontología original de partida[4]. Dado el limitado vocabulario presente en la ontología, no se podía mapear directamente ésta con la mayoría de las palabras presentes en los documentos. Para salvar este inconveniente se utilizó una base de datos lingüística externa (WordNet) la cual permitió enriquecer los conceptos de la ontología. De esta forma se logró extender cada uno de los conceptos de la ontología con palabras obtenidas de WordNet. Para realizar la extensión de los conceptos los autores se enfrentaron con una serie de problemas que se describen de forma más detallada en el Anexo A, *Estudio del Estado del Arte*.

Básicamente el sistema que fue desarrollado está formado por tres componentes: el módulo de extracción, el módulo de mapeo, y el módulo de optimización. La entrada

[3]En los anexos A.1.3 y A.1.4 se desarrollan dos trabajos que hacen uso de estas técnicas.
[4]La ontología de partida utilizada fue: "*Web Yahoo*".

del sistema es un documento web y la salida es un conjunto de conceptos, los cuales refieren al tema del documento.

El *módulo de extracción* es el encargado de manejar el proceso de la obtención de las oraciones más relevantes del documento. El sistema de extracción diseñado está basado en tags HTML. Esto es así porque los autores creen que algunos de los tags HTML indican la ubicación de ideas que el autor del documento desea destacar (por ejemplo el tag TITLE). Los autores también reconocen que existen documentos que no contienen tags HTML, documentos no estructurados, pero que estos serían considerados en un trabajo futuro, y no será adecuada la utilización de este mecanismo para estos documentos.

Luego de la extracción de los conceptos destacados, se realizan dos tareas fundamentales para poder relacionar semánticamente los conceptos y poder realizar el mapeo de forma adecuada. Estas tareas son la de *"stemming"*[5] y *"sense-tagging"*[6]. La primera intenta convertir la palabra a su raíz y la segunda intenta determinar el significado de la palabra.

En el *módulo de mapeo* las palabras clave son relacionadas con los conceptos de la ontología Yahoo, que tiene conceptos *"stemmed"* y *"sense-tagged"*, es decir tienen el mismo formato que los conceptos de entrada. Aquellas palabras clave que no se pudieron mapear con la ontología se tratan de mapear con la ontología extendida.

Para poder identificar los conceptos más importantes, se asocia a cada concepto un peso. Este peso a grandes rasgos es calculado a partir de dos variables, una es el tipo de mapeo (normal o extendido) y la otra es la frecuencia de aparición de un concepto en el documento. Es necesario considerar el tipo de mapeo debido a que en la ontología extendida la probabilidad de error aumenta, dado que hubo dos procesos de *"stemmed"* y *"sense-tagged"* (uno para extender la ontología y otro para mapear la palabra clave).

El *módulo de optimización* permite reducir el árbol de la ontología de forma que sólo los conceptos activos e intermedios (entre conceptos activos) estén presentes. El árbol de la ontología reducido es convertido a un único camino con los nodos más altos (ponderación mixta) usando el algoritmo de *Maximal Spanning Tree*. Luego de esto se elige el concepto más relevante para asignarlo como título del documento.

Los autores probaron empíricamente que el 57.8 % de las palabras pudo ser mapeado directamente en el concepto correspondiente a la ontología y un 17.8 % en los conceptos extendidos.

En el proyecto *Knowledge-based Automatic Topic Identication*[23], de Chin-Yew Lin, Departamento de Ingeniería Eléctrica, Universidad del Sur de California, Los

[5]Stemming, es un método por el cual se obtiene la raíz de una palabra, este método es usado en sistemas de recuperación de información permitiendo aumentar el *recall*. Por ejemplo la palabra *"bibliotecario"* tiene como raíz la palabra "*bibliotec*", o sea si se está buscando la palabra *"biblioteca"* usando el método de stemming, no sólo se recuperaran aquellos documentos que contengan la palabra biblioteca sino que también se recuperaran los documentos que tengan la palabra *"bibliotecario"* ya que ambas palabras tienen la misma raíz.

[6]Sense-tagging, es la asignación automática del significado léxico de las palabras de un texto.

Angeles, Estados Unidos, nuevamente encontramos el uso de WordNet para poder representar y relacionar semánticamente los conceptos del documento. Básicamente se plantea un método para encontrar el tema central del texto, basado en identificar y ponderar los conceptos. Para representar conceptos abstractos (generales) utilizaron WordNet. Además presenta el método de conteo de palabras y propone el método de conteo de conceptos.

Con el objetivo de contar la frecuencia de los conceptos, emplearon una generalización jerárquica de conceptos, construida a partir del texto (corpus) y WordNet. Utilizando la jerarquía, se enfrentaron al problema de encontrar la generalización más apropiada. Claramente no se pueden usar las hojas ya que en este nivel no se ha adquirido el poder de generalización suficiente, es decir la abstracción suficiente. Tampoco se podrá utilizar el nivel más alto, nos vamos a lo muy general. Consideramos que este proceso puede ser de gran valor para determinar que tan cercana es la relación semántica entre la entrada proporcionada por el usuario de nuestro sistema y fragmentos del corpus (documento) a analizar. Como solución al problema anterior proponen *"ratio de frecuencia de conceptos"* y *"starting depth"*. Definieron la *frecuencia de ocurrencia del concepto C y sus subconceptos* como el peso del concepto, es decir la *frecuencia del concepto C* es la suma de la frecuencia de los conceptos hijos más la frecuencia del concepto padre. Se puede observar que un concepto más general (el concepto padre de otro) tiene peso mayor o igual al máximo de todos los pesos de los hijos.

El *ratio (R)* de cualquier concepto C lo definieron mediante la siguiente fórmula:

$$Ratio\,(C) = \frac{\text{máx}(peso\,de\,todos\,los\,hijos\,de\,C)}{\sum(peso\,de\,todos\,los\,hijos\,de\,C)}$$

R es una manera de identificar el grado de síntesis del concepto (nodo). Podemos observar que un concepto con un ratio pequeño referencia a sus hijos de forma más equitativa, si nos quedamos con el hijo perdemos mucha información relevante, mientras que un concepto con alto ratio, cercano a 1, tiene un concepto hijo muy referenciado en el texto, muy destacado. Si un concepto tiene un ratio alto, quiere decir que no generaliza los conceptos hijos. Esto implica que existe sólo un concepto hijo o pocos, o la suma de los pesos de los hijos está concentrada en un único concepto (hijo).

Con el objetivo de elegir el concepto con la apropiada generalización utilizaron un parámetro *"starting depth"*. Se determinó que el criterio de ratio sea aplicado (tenga efecto) después de que se haya excedido el *"starting depth"* como profundidad en la jerarquía. La primera colección de nodos interesantes luego de pasada la profundidad será la colección seleccionada.

Un trabajo similar al anteriormente mencionado es el correspondiente al proyecto *RICOTERM-2*[19], el cual presenta un sistema de RI mediante la expansión de consultas. Una característica que se destaca de este proyecto es que fue implementado para funcionar en múltiples idiomas, entre ellos el español. En este trabajo se utilizaron dos estrategias textuales complementarias para la expansión de consultas: el

uso de una herramienta de extracción automática de términos y el uso de un detector de relaciones conceptuales (por ejemplo sinónimos, hiperónimos e hipónimos), que permitan identificarlos y etiquetarlos en su contexto. Además se utilizó una base de datos terminológicos y de diccionarios especializados sobre economía, que incluyen definiciones. Los autores utilizaron EuroWordNet para expandir los términos de la consulta.

En el proyecto *Identifying Topics by Position*[28] de Chin-Yew Lin, Eduard Hovy, Instituto de Ciencias de la Información de la Universidad del Sur de California, se presenta información importante sobre cómo están conformados los documentos (artículos técnicos y papers) y cómo se puede reducir considerablemente el área de búsqueda de las palabras clave útiles para la identificación del tema del documento. En sus conclusiones afirman que en la mayoría de los documentos se puede obtener el tema analizando la primera oración del primer párrafo y la penúltima oración del último párrafo.

Este proyecto presenta información importante sobre cómo están conformados los documentos y cómo se puede reducir drásticamente el área de búsqueda de las palabras claves útiles para la identificación del tema del documento. Aunque consideramos que el método descripto, así como los resultados de los autores, no se aplican directamente al objetivo de nuestro proyecto, alguna variación de esta técnica podrá servir como apoyo.

Otro trabajo interesante es *Using frequently occurring words to identify the subject of a document*[25] realizado por Oer Drori de la Universidad Hebrew de Jerusalén. Drori presenta un mecanismo para determinar automáticamente la temática de un texto, utilizando un método consistente en leer el documento y utilizar un análisis estático (sin considerar relaciones semánticas) para determinar las palabras con más frecuencia en el documento. El sistema se basa en datos estadísticos pero no tiene en cuanta fenómenos lingüísticos. Este método presenta una debilidad importante que es que el resultado está sujeto a la cantidad de palabras del texto (tamaño del documento), es decir, es más efectivo con textos grandes, dado que se puede encontrar más ocurrencias de una misma palabra. El autor se encontró con la dificultad de que las palabras gramaticales como *and*, *of*, *or*, ocurren muchas veces en un texto, por lo tanto este tipo de palabras no las consideraron para catalogar el documento. Para esto el texto es analizado luego que las palabras sin contenido léxico (funcionales o gramaticales) y que no aportan información para la determinación del tema del documento son descartadas usando una lista de palabras (*"Stop List"*).

El procesamiento lingüístico limitado tiene la tarea de reconocer las inflexiones[7] en el lenguaje y además las palabras insignificantes (sin sentido léxico) que serán descartadas por una *Stop List* que es específica para cada lenguaje.

[7] Cada uno de los cambios morfológicos que sufren las palabras sujetas a flexión. Por ejemplo, los verbos castellanos tienen 6 inflexiones verbales básicas, que antiguamente correspondían a las tres personas del discurso, tanto en singular como en plural. Por ejemplo las inflexiones de la palabra *"hablar"* son: yo *"hablo"*, tú *"hablas"*, él *"habla"*, nosotros *"hablamos"*, vosotros *"habláis"*, vos *"hablás"*, ellos *"hablan"*.

Al depender del tamaño del documento y no tomar en cuenta las relaciones semánticas este mecanismo no va a ser de mucha utilidad para cumplir nuestros objetivos que están fuertemente ligados a poder tener el conocimiento de lo que se está expresando en el documento y por ende tomar en cuenta la semántica de las palabras y oraciones. Quizá sea de utilidad como un mecanismo de apoyo para el algoritmo principal, es decir, identificar relevancias en el documento. Algo a destacar es el uso de la lista *Stop List* como solución para palabras que aparecen de forma repetida en el documento y que no se tendrían que tomar en cuenta para la determinación del tópico.

El trabajo *Algoritmo de resolución de la anáfora pronominal en diálogos*[29] para el castellano realizado por Patricio Martínes-Barco, de la Universidad de Alicante, presenta un algoritmo que combina dos aproximaciones diferentes. Por una parte, hace uso de información lingüística para aceptar o rechazar el antecedente de la anáfora mientras que por otra parte, usa información derivada de la estructura del diálogo para proponer dicho antecedente. Algunos aspectos interesantes de destacar son la descomposición en componentes, las diversas fuentes de información (acá volvemos a encontrar la aparición de WordNet como recurso léxico), la dificultad que encontraron los autores para resolver algunos problemas y la forma en cómo fusionaron la información de la estructura del documento con la información lingüística y como esta integración permitió mejorar los resultados. Por más información ver Anexo A.1.8.

3.4. Formas de evaluación

En esta sección se presentarán mecanismos utilizados para la evaluación de los sistemas de RI y de EI.

3.4.1. Evaluación de Sistemas de Recuperación de Información

Método de evaluación de buscadores web

Un método para la evaluación de un conjunto de buscadores web[33] presentado por Marta Torres Magán de la Universidad Carlos III de Madrid, consta de los siguientes pasos:

- **Selección de los buscadores**: En primer lugar escogeremos los buscadores a evaluar, por ejemplo Google y Yahoo.
- **Selección de las consultas**: La evaluación se realizará en base a una batería de consultas. Resulta crítica la elección de las mismas, pues de ellas dependerá el resultado de la evaluación. Se recomienda escoger consultas que abarquen las siguientes posibilidades:
  - Consultas con una única palabra no ambigua.
  - Consultas con varias palabras.

    - Consultas con varias palabras, de las cuales al menos una resulta por sí misma ambigua pero tiene sentido en el contexto de la consulta.

- **Realizar las consultas**: Se deberá realizar la batería de consultas anterior contra todos los buscadores web sujetos a evaluación, anotando los resultados para su posterior análisis.

- **Evaluación de resultados de las consultas**: Cada página recibirá un peso según su relevancia con la consulta realizada. Para ello se empleará la siguiente escala de calidad:

    - 2: Lo que se buscaba. Cuando el resultado obtenido coincide con los esperado.
    - 1: Casi. Cuando la página obtenida contiene parte de lo esperado.
    - 0: Irrelevante. Cuando la página obtenida no coincide con lo esperado.
    - -1: Enlace roto o duplicado. Cuando no se puede acceder a la página obtenida.

  Como el orden en el que aparecen las páginas en las primeras posiciones de la consulta resulta muy importante, los pesos anteriores serán multiplicados por el inverso de la posición que ocupan (1/posición). Según lo anterior, si la primera página coincide con lo esperado aportarán 2 puntos (2/1), mientras que si la cuarta página ofrece parte de la información buscada contará a la evaluación con 0.25 puntos (1/4).

- **Otros factores**: Si la evaluación anterior ofrece resultados muy parecidos (difieren en menos de un 10 %) será cuando cobren importancia otros factores de evaluación:

    - **Factores objetivos**: Número de páginas contenidas en el buscador web y rapidez de los resultados.
    - **Factores subjetivos**: Accesibilidad al motor de búsqueda y otros servicios ofrecidos. La forma de emplear los aspectos anteriores puede variar según los objetivos de la evaluación y el público a la que va dirigida.

3.4.2. Evaluación de Sistemas de Extracción de Información

Método de evaluación de un sistema de extracción de información genérico[21].

Los sistemas de EI se pueden ver como un algoritmo que recibe como entrada un documento y brinda como resultado un conjunto de tuplas de relaciones. Por ejemplo, un sistema de extracción de información puede procesar los artículos de un diario e intentar identificar uniones entre empresas (*Empresa 1* compró *Empresa 2*) o eventos de sucesión (*Persona 1* sucede a la *Persona 2* como vicepresidente de la empresa).

Como es de esperar, tales sistemas son inherentemente ruidosos y generan salidas imperfectas. Algunas veces tuplas erróneas aparecen en documentos y a veces generan tuplas espurias. Una de las cuestiones más importantes es cómo poder evaluar estos sistemas objetivamente y con la mínima cantidad de esfuerzo.

Una estrategia común es el uso de las curvas de *precision* y *recall*, que muestran cómo el sistema se comporta bajo diferentes configuraciones. La *precision* es una medida de correctitud de la información recuperada por el sistema, se define como el número de tuplas correctas sobre el total de tuplas generadas. El *recall* es una medida de cobertura sobre la cantidad de información relevante recuperada, se define como el número de tuplas correctas recuperadas sobre el total de tuplas que podrían haber sido extraídas del documento.

$$Precision = \frac{tuplas\,correctas}{tuplas\,obtenidas}$$

$$Recall = \frac{tuplas\,correctas}{tuplas\,identificables}$$

Notar que la *precision* y el *recall* son antagónicos entre si, por ejemplo un sistema que recupere el documento completo tendrá un *recall* del 100 % pero un *precision* muy baja. Esta situación genera la necesidad de utilizar una fórmula de balance que combine las medidas de *precision* y *recall*. Una medida de balance de uso frecuente es *F-measure*.

$$F - measure = \frac{(\beta^2 + 1)\,precision.recall}{\beta^2 precision + recall}$$

Cuando β es 1, la *precision* y el *recall* tienen el mismo peso. Un valor de β mayor a 1 implica que la *precision* tendrá mayor peso, y cuando β es menor que 1 el *recall* es beneficiado.

Para obtener los valores de *precision* y *recall* es necesario ejecutar el sistema para un conjunto de documentos y comparar el conjunto de tuplas propuestas por el sistema con la identificación realizada manualmente.

4 Solución propuesta

En esta sección se presenta la solución propuesta para alcanzar los objetivos planteados.

Luego de realizado el estudio del estado del arte, no se encontró ningún trabajo que se enfrentara al problema de la identificación temática, o más precisamente, la identificación de segmentos o porciones de un documento relacionados con una expresión de consulta, y le diera una solución satisfactoria. En cambio, se encontraron varios trabajos teóricos que especifican de cierta forma las diferentes tareas o etapas que deben realizarse, y también se encontraron varios trabajos que se enfrentaron a problemas similares. Tomando como base información recopilada de estos trabajos, logramos plantear una solución que permitió alcanzar los objetivos del proyecto.

4.1. Flujo de componentes

En primera instancia comenzaremos por presentar la arquitectura propuesta. La misma fue estructurada considerando los principales requerimientos y obstáculos que se debieron resolver. Siguiendo los principales lineamientos de los sistemas de EI estudiados, se organizó la arquitectura en componentes independientes, cada uno encargado de resolver un problema específico del sistema. Para cada componente se plantearon variadas alternativas y se analizaron las ventajas y desventajas de cada una, seleccionando en cada caso la alternativa que en conjunto proporcionan la solución más adecuada. Esta discusión se describe completamente en el capítulo 5 *Desarrollo del Proyecto.*

Los principales componentes que integran la solución son: *"Tokenizador, Analizador Morfológico y Semántico*", *"Analizador de Relaciones",* e *"Identificador de Segmentos Relacionados"*. A continuación presentaremos cada componente describiendo las entradas y salidas que tiene cada uno, así como las tareas que realizan.

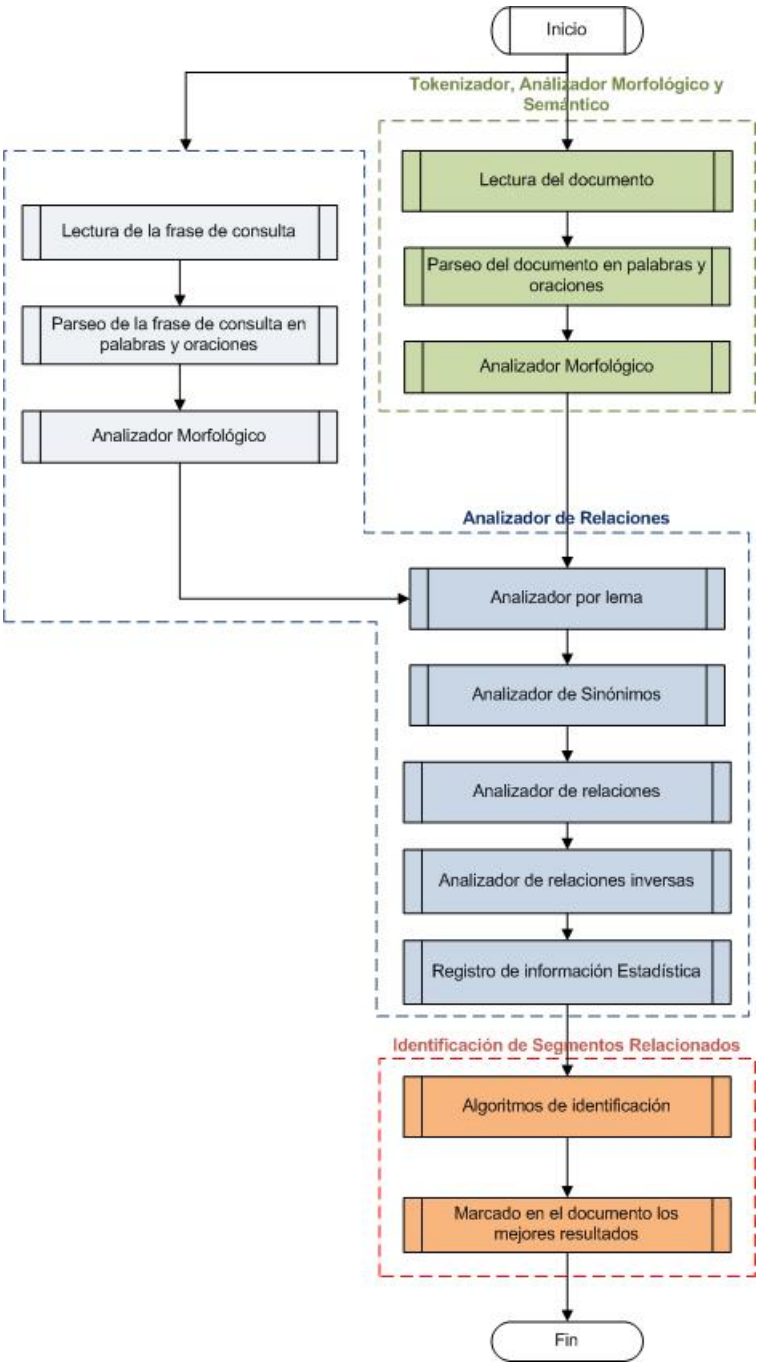


Figura 4.1: Esquema de componentes de la solución

4.1.1. Tokenizador, Analizador Morfológico y Semántico

Este es el componente encargado de realizar la tokenización y análisis morfológico de los documentos. El componente tiene como entrada un documento y devuelve un conjunto de anotaciones con la información de la estructura del mismo. Tal informa-

ción está formada por el conjunto de las palabras, locuciones, entidades con nombre, o términos del documento, cada uno con su correspondiente información morfológica representada mediante una etiqueta, su lema, sus significados y ubicación dentro del documento.

Para mostrar cómo debería ser el comportamiento del componente se presentará en el Cuadro 4.1 una oración y el resultado esperado del procesamiento de ésta.

"Les di bananas a los monos porque ellos tenían hambre."

Frase	Les	di	bananas	a	los	monos	porque	ellos	tenían	hambre	.
Lema	le	dar	banana	a	el	mono	porque	ellos	tener	hambre	.
Etiqueta	VMIS1S0	NCFP000	SPS00	DA0MP0	NCMP000	CS	PP3MP000	VMII3P0	VMII3P0	NCFS000	Fp
ID Significado	01185771				03069005			01508689	01794357	10112984	

Cuadro 4.1: Ejemplo del procesamiento de la frase tomado de Freeling[1].

Este componente también identifica las oraciones del texto y registra información estadística del mismo, como son la cantidad de nombres, cantidad de nombres con significado, verbos, pronombres, entre otros.

En resumen, el tokenizador separa el texto en palabras y el analizador morfológico agrega información sobre éstas que luego será analizada por los siguientes componentes.

4.1.2. Analizador de Relaciones

Este componente cumple con la función de identificar las relaciones semánticas que existen entre las palabras de la frase de consulta del usuario con las palabras que forman parte del documento procesado. El componente tiene como entrada un documento, la frase de consulta y el análisis del documento realizado por el componente anterior, y tiene como salida un conjunto de relaciones entre la consulta y el documento. Son muchas las relaciones analizadas, como por ejemplo sinónimos, hiperónimos, hipónimos y distintos tipos de merónimos (el conjunto completo de las relaciones identificadas puede verse en el Capítulo 5 *Desarrollo del Proyecto*).

Además se determinan si existen relaciones mediante la comparación de lemas, es decir, se comparan los lemas de la frase de consulta con los de las palabras del documento; estos lemas deben tener la misma categoría gramatical. Un punto importante a destacar es que se utilizó un mecanismo para filtrar palabras (por ejemplo pronombres) semejante al *Stop List* presentado en la sección 3.3.1 del *Marco Teórico*. Para estas palabras filtradas no tiene sentido comparar sus lemas pues esa relación no tendría valor para nuestro sistema.

Por ejemplo, dada la frase de entrada *"Los animales viven en el planeta"* y el texto *"En la Tierra existen numerosos seres vivos."*, el analizador de relaciones debe ser

capaz de relacionar como hiperónimos a las palabras *"animales"* y *"seres vivos"* y debe relacionar como sinónimos a las palabras *"viven"* y *"existen"* y *"planeta"* con *"Tierra"*.

4.1.3. Identificador de Segmentos Relacionados

El tercer componente se encarga de procesar por medio de distintos algoritmos la información recopilada por los módulos antes presentados. El objetivo principal es seleccionar en el documento las oraciones que se relacionan de mejor forma con la frase de consulta.

Se pretende que el diseño del identificador sea flexible de forma que se pueda especificar el algoritmo que realice el análisis de la información y también que permita la configuración de ciertos parámetros, como por ejemplo los pesos de las relaciones y la profundidad máxima de búsqueda con el fin de determinar relaciones recursivas.

Se desarrollaron tres algoritmos con distintas características. A grandes rasgos, los algoritmos propuestos consisten en ponderar las distintas relaciones identificadas y por medio de la valoración de las oraciones, seleccionar aquellas que tienen un mayor valor. Al estudiar numerosas relaciones y muchas de ellas recursivas en varios niveles, surge la necesidad de poder especificar que algunos tipos de relaciones son más importantes que otras. Por ejemplo, si la consulta tiene la palabra *"animal"* y una oración contiene las palabras *"ser vivo"* y otra la palabra *"entidad",* el peso de la primera oración debiera ser superior al de la segunda, ya que *"ser vivo"* es un sinónimo directo de la palabra de la consulta, mientras que *"entidad"* es hiperónimo recursivo de un nivel alto.

Resumiendo, una relación identificada entre el documento y la frase de consulta tiene una ponderación determinada por el valor de la relación multiplicado por un coeficiente entre 0 y 1 que es asignado según la profundidad de la misma.

Los distintos criterios de selección de resultados se presentan en la sección 5.4.1

4.2. Problemas enfrentados y mejoras futuras

Para finalizar la presentación de la solución, brindaremos un breve resumen enumerando por un lado aquellos problemas que se abordaron en la solución propuesta, así como también aquellos identificados, y que por falta de tiempo o recursos, se dejan para ser resueltos en proyectos futuros.

Algunos de los problemas resueltos en la solución son:

1. Tokenización del texto en palabras y reconocimiento de oraciones.
2. Identificación de entidades con nombre, fechas y números.
3. Reconocimiento de locuciones y palabras compuestas.

4. Reconocimiento del *"Part of the Speach"* (categoría gramatical) de las palabras . Identificación de nombres, verbos, adverbios y adjetivos.

5. Reconocimiento del *synset*[1] o *synsets* al que pertenece una palabra.

6. Identificación del lema de una palabra.

7. Búsqueda de los sinónimos de una palabra.

8. Reconocimiento de relaciones semánticas como hiperónimos, hipónimos, merónimos, holónimos, sinónimos, entre otras.

9. Expansión de la frase de consulta del usuario.

10. Desambiguación del significado de las palabras por métodos estadísticos, automáticos y a través de la interacción con el usuario.

11. Análisis de completitud de cada oración con la frase de consulta.

12. Utilización de una lista de palabras sin sentido (*"Stop List"*) para ignorar palabras que no agregan información semántica.

13. Identificación de segmentos del documento relacionados con la frase de consulta.

Algunas de las problemáticas propias del área que no fueron desarrolladas en la solución propuesta son:

1. Identificación de la forma raíz de una palabra y análisis de familias léxicas. La identificación de los lemas de las palabras puede suplir en parte la necesidad de realizar este análisis.

2. Análisis de metáforas y metonimias. El análisis de metáforas y metonimias está todavía lejano, debido a la necesidad de poder brindarle al sistema información del mundo. Sí se podría buscar un acercamiento a través de la utilización de algún diccionario de metáforas.

3. Identificación del tema del documento. Aunque se estudiaron diferentes mecanismos para lograr identificar el tema de un documento, realizar este análisis no se consideró necesario para alcanzar la solución. A esto se suma el hecho de que los mecanismos que lograron mejores resultados realizan un procesamiento de relaciones del documento completo, lo que es computacionalmente muy costoso.

[1]Cada sysnset representa un concepto distinto. WordNet agrupa los nombres, verbos, adjetivos, adverbios en sinónimos cognitivos (sysnsets), los cuales están relacionados o interconectados mediante el significado semántico-conceptual y relaciones léxicas.

4. Soportar la existencia de errores ortográficos, de capitalización y de puntuaciones. Si bien se podrían realizar mecanismos para evitar que estos errores en el documento afecten el funcionamiento del sistema, esto se dejó fuera del alcance y la identificación se basa en el supuesto de que los documentos son ortográficamente correctos, la capitalización de las palabras es correcta y los signos de puntuación son correctos.

5. Análisis de correferencias y reconocimiento de anáforas. El análisis de correferencias es una de las tareas más complicadas dentro de los sistemas de extracción de información. Si bien se estudiaron algunos métodos que solucionan el problema para algunos casos específicos, el desarrollo de una solución general fue dejado para futuros proyectos.

6. Identificación de palabras en idiomas diferentes al español. Si bien estamos conscientes de que muchos textos en español contienen palabras en inglés, éstas no son procesadas por nuestro sistema, el cual está especializado para el idioma español.

5 Desarrollo del proyecto

El objetivo de este capítulo es explicar como se implementó la solución teórica presentada en el capítulo anterior. Comenzaremos mostrando el conjunto de herramientas y tecnologías utilizadas, brindando detalles de cada una. También se presentará la arquitectura del sistema ISRT. Luego se analizarán las principales dificultades de implementación que se enfrentaron y por último se detallarán las decisiones de implementación más importantes.

Se espera que al final del capítulo se pueda comprender cómo fue desarrollado el proyecto y sus dificultades.

5.1. Herramientas y tecnologías seleccionadas

En esta sección se presentan el conjunto de las herramientas analizadas y finalmente utilizadas durante el desarrollo del proyecto. Se detallan las características propias de cada una y los motivos por los cuales fueron elegidas.

Considerando el análisis del problema planteado y con el soporte del estudio del estado del arte, llegamos a la conclusión de que para alcanzar la solución necesitaríamos utilizar herramientas que nos brindaran apoyo en los siguientes puntos:

- Documentar el proyecto.
- Administrar la información.
- Implementación.
- Tokenizar los documentos.
- Expandir la consulta.
- Relacionar la consulta con el documento.
- Integrar nuestros componentes con Lavinia.

Para cumplir con los puntos antes mencionados reunimos el siguiente conjunto de herramientas: LyX, Google Code + SVN Tortoise, Java, MySQL, Apache Tomcat, Eclipse y especializadas en el procesamiento del lenguaje las herramientas UIMA, Lavinia, Freeling y Spanish WordNet. La utilización de este conjunto compatible de herramientas permitió el desarrollo del sistema.

A continuación presentaremos las herramientas Freeling, Spanish WordNet y UIMA, las cuales consideramos claves para la realización del proyecto, indicando para

cada una el motivo de su selección. El resto de herramientas utilizadas en el desarrollo del sistema se presentan en el Anexo D, *Otras Herramientas*.

5.1.1. Freeling

Freeling consiste en un conjunto de librerías que proveen servicios para el análisis lingüístico. Ha sido diseñado para ser usado como una librería externa para cualquier tipo de aplicación que requiera esta clase de servicios. También contiene un programa ejecutable capaz de ofrecer una interfaz básica para la interacción con los recursos provistos.

Dentro de los principales servicios brindados por Freeling se encuentran: un tokenizador de textos, un divisor en oraciones, un analizador morfológico, tratamientos de sufijos, reconocimiento de palabras múltiples y locuciones, predicción probabilística de palabras de categorías desconocidas, reconocimiento de entidades con nombre, reconocimiento de números, fechas. También cuenta con *POS Tagging* y brinda anotaciones semánticas basadas en WordNet. A estos servicios y otros no nombrados se suma que Freeling está disponible para varios idiomas como el español, catalán, italiano e inglés.

Una de las características interesantes es que para cada palabra analizada, Freeling retorna sus posibles significados (anotaciones semánticas), éstos son tomados de labase de datos de Spanish EuroWordNet, lo que apoya la utilización de estas herramientas en conjunto, una encargándose de la tokenización e identificación de las entidades con nombre, y la otra encargándose de determinar las relaciones. Esta posibilidad de combinación fue una de las principales razones de utilizar este conjunto de herramientas. En realidad no se utilizó EuroWordNet, sinó Spanish WordNet la cual será presentada a continuación y está incluida en la primera.

5.1.2. Spanish WordNet

WordNet es una gran base de datos léxica de inglés, desarrollada bajo la supervisión de George A. Miller Figura 5.1 en la Universidad de Princeton.

Figura 5.1: George A. Miller, director y principal responsable de WordNet

En WordNet los nombres, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos cognitivos llamados *synsets*, cada uno expresando un concepto. Los *synsets* en WordNet están interconectados por medio de relaciones conceptuales y léxico-semánticas. Su homólogo europeo, EuroWordNet, es una base de datos multilingüe con WordNet's para varios idiomas (holandés, italiano, español, alemán, francés, checo y estonio). EuroWordNet está estructurado de la misma manera que la versión americana en términos de *synsets* con relaciones semánticas básicas entre ellos. El componente en español del EuroWordNet es el *Spanish WordNet*.

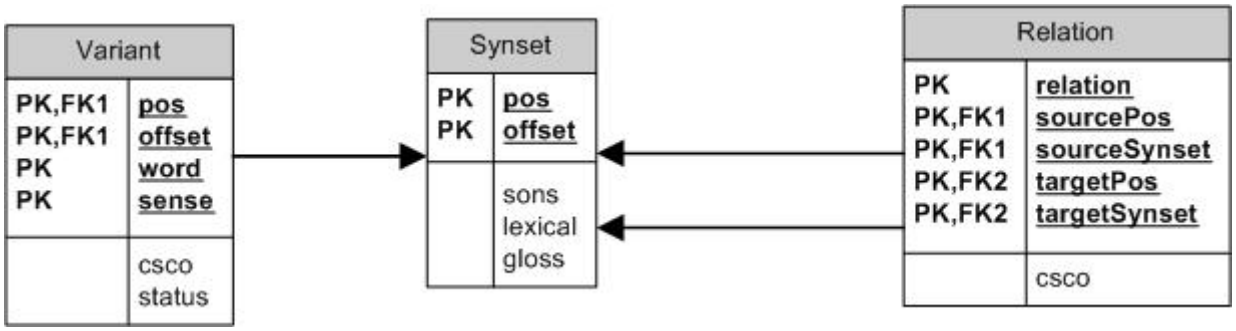


Figura 5.2: Modelo Entidad Relación de la base de datos WordNet

La base de datos de Spanish WordNet está compuesta por 105.516 *synsets*, 93425 *variants* y 145.670 *relations*. Las relaciones se dividen en distintos tipos, ver Cuadro 5.1 en el cual se presentan ejemplos para cada una.

Spanish WordNet es una parte clave del proyecto, la información que provee fue utilizada en la expansión de la consulta en sinónimos y otros tipos de relaciones, así como también en el reconocimiento de relaciones entre las palabras de la frase de consulta y las palabras del documento. Con esta herramienta pudimos encontrar en el documento referencias a elementos de la consulta que no son identificables directamente, permitiendo aumentar enormemente los resultados encontrados.

Dada la cercana relación entre Freeling y WordNet, la ventaja del hecho que en el resultado del procesamiento de Freeling obtengamos los *offsets*[1] de las palabras (mismos *offsets* utilizados en WordNet), favorece mucho la utilización de estas herramientas en conjunto.

[1]Son un conjunto de identificadores para los posibles significados de una palabra. En la base de datos WordNet el *offset* sirve para identificar un *synset*. Más específicamente un *offset* y una categoría gramatical determinan un *synset* (concepto). Ver Anexo A.2.1.

Relación	Cantidad	Descripción	Ejemplo
has_hyponym	84048	Tiene un Hipónimo	*taxi / auto*
near_synonym	21887	Tiene un quasi-sinónimo	*abreviado / acortado*
has_holo_member	11848	Tiene un holónimo tipo miembro-colección	*flota / barco*
near_antonym	7453	Tiene un quasi-antónimo	*bueno / malo*
has_holo_part	6881	Tiene un holónimo tipo componente-objeto	*mano / dedo*
has_derived	6052	Tiene un derivado	*eolio / viento*
see_also_wn15	3286	Ver también	*dormitar / dormirse*
has_holo_madeof	709	Tiene un holónimo tipo substancia-objeto	*libro / papel*
be_in_state	652	Está en el estado	*honorabilidad / honorable*
role_agent	516	Papel-Agente	*fabricar/fabricante*
has_xpos_hyponym	482	Tiene un hipónimo de distinta categoría gramatical	*franqueza / abierto*
has_subevent	427	Tiene sub evento	*enjabonarse / lavarse*
xpos_near_synonym	321	Tiene un quasi-sinónimo de distinta categoría gramatica	*prisa / acelerar*
role_instrument	291	Papel-Instrumento	*carpintero / martillo*
verb_group	262	Verbo-Grupo	*simular/actuar*
causes	243	Causas	*relajar / mitigar*
role	106	Papel	*flete / transportar*
role_location	83	Papel-Localización	*carpintero / carpintería*
pertains_to	50	Pertenece a	*oral / boca*
has_mero_part	28	Tiene un merónimo tipo componente-objeto	*dedo / mano*
role_patient	6	Papel-Pasivo	*inquilino / alquilar*
has_mero_member	2	Tiene un merónimo tipo miembro-colección	*barco / flota*

Cuadro 5.1: Conjunto de relaciones presentes en Spanish WordNet

5.1.2.1. Sinónimos

Los sinónimos determinan la organización de la base de datos en *synsets*. En WordNet todas las palabras semánticamente equivalentes deben pertenecer al mismo *synset*.

Una definición formal de sinónimos[**?**] dada por Leibniz[2] es:

"Dos expresiones son sinónimos si la sustitución de una de ellas por la otra ***nunca*** *cambia el valor de verdad de la oración en la cual fue realizada la sustitución"* Sin embargo, los verdaderos sinónimos son raramente encontrados en el lenguaje. Por esto Miller[3] y Fellbaum[4], sugirieron en 1990 una definición más débil, denominada *"similaridad semántica"*, ésta es:

"Dos expresiones son sinónimos en un contexto lingüístico ***C*** *si la sustitución de una de las expresiones por la otra en el contexto* ***C*** *no altera el valor de verdad"*

5.1.2.2. Antónimos

Los *antónimos* se refieren a léxicos opuestos, como por ejemplo *"subir"* y *"descender"*, *"bueno"* y *"malo"* o *"justicia"* e *"injusticia"*. Es evidente que es una relación simétrica, pero poco más se puede decir, ya que parece codificar una gran variedad de fenómenos de la oposición, por ejemplo, *"ricos"* y *"pobres"* son opuestos escalares con muchos valores entre los extremos, *"muertos"* y *"vivos"* puede verse como opuestos complementarios.

Relación WordNet: NEAR_ANTONYM

Aplica sobre: verbos, nombres, adjetivos, adverbios.

5.1.2.3. Hipónimos

WordNet se ha construido en torno a las relaciones de *hipónimos*, lo que las convierte en unas de las relaciones más importantes. Cadenas de relaciones tales como:

---

[2]Gottfried Wilhelm von Leibniz (Leipzig, 1 de julio de 1646 - Hannover, 14 de noviembre de 1716) fue un filósofo, matemático, jurista, bibliotecario y político alemán.
http://es.wikipedia.org/wiki/Gottfried_Leibniz

[3]George Armitage Miller (Nació 3 de Febrero de1920 en Charleston, West Virginia), es el autor de algunos de los papers de psicología citados con más frecuencia, *The Magical Number Seven, Plus or Minus Two* publicados en 1956. También es conocido por haber sido fundador de WordNet una enorme base de datos léxica y el sistema Simpli que permite realizar búsquedas en internet desambiguando la consulta.
http://en.wikipedia.org/wiki/George_Armitage_Miller

[4]Christiane D. Fellbaum, nacida en Braunschweig, Lower Saxony, Alemania, ha vivido en Estados Unidos desde 1969. Despues de su graduación en la Universidad de Pricenton con un PhD en lingüística, ella pasó a formar parte del Departamento de Ciencias Cognitivas bajo la supervición de George A. Miller y desde ese momento ha jugado un rol activo en el desarrollo de WordNet. Ella es la fundadora y presidente de la Asociación Mundial de WordNet (Global WordNet Association), cuya meta es la construcción de múltiples bases de dátos lexicas para muchos lenguajes alrededor del mundo.
http://en.wikipedia.org/wiki/Christiane_Fellbaum

entidad HAS_HYPONYM *objeto* HAS_HYPONYM *instrumento*
HAS_HYPONYM *vehículo* HAS_HYPONYM *vehículo de motor*
HAS_HYPONYM *coche* HAS_HYPONYM *taxi*

pueden constituir la columna vertebral de una base de conocimiento léxico, a través de especificaciones que pueden ser heredadas de una manera coherente a miles de conceptos más específicos.

Relación WordNet: HAS_HYPONYM
Aplica sobre: nombres y verbos.

5.1.2.4. Merónimos

Los *merónimios* son relaciones *parte-todo,* están divididos en las siguientes sub categorías :

- HAS_HOLO_MEMBER / HAS_MERO_MEMBER, entre un conjunto y sus miembros (por ejemplo, *"flota" - "barco"*).
- HAS_HOLO_PART / HAS_MERO_PART, entre un conjunto y sus partes constituyentes (por ejemplo, *"mano" - "dedo"*).
- HAS_HOLO_MADEOF entre un elemento u objeto, y el material por del que está hecho (por ejemplo, *"libro" - "papel"*).

Aplica sobre: nombres.

5.1.2.5. Rol e Involucrados

Desde un punto de vista cognitivo, la función *Rol e Involucrados* es una de las principales características que organiza el conocimiento humano. Del mismo modo, la funcionalidad está ampliamente reflejada en el léxico. Las lenguas son ricas en procedimientos de derivación que generan los nombres de los verbos, como por ejemplo, *correr y corredor*, de *teléfono y telefonear*. Este tipo de relaciones se divide en las siguientes relaciones más específicas:

- ROLE_INSTRUMENT, aplica sobre: nombre-nombre, nombre-verbo.
- ROLE_AGENT, aplica sobre: nombre-nombre, verbo-nombre.
- ROLE_PATIENT, aplica sobre: nombre-nombre, nombre-verbo.
- ROLE_LOCATION, aplica sobre: nombre-nombre, verbo-nombre, verbo-adjadv, nombre-adjadv.

La relación ROLE se utiliza para los casos en que las pruebas o los criterios para la extracción de recursos de estas relaciones no pueden discriminarse entre los subtipos anteriormente mencionados. Aplica sobre nombre-verbo, nombre-nombre.

5.1.3. UIMA

La utilización de UIMA *(Unstructured Information Management Applications)* fue indispensable, sobre todo para lograr la integración con Lavinia, por eso desde un principio su utilización no fue cuestionada. Pero si la integración con Lavinia no hubiera sido requerida, la utilización de UIMA igualmente hubiera sido necesaria. Esto se debe a muchas ventajas que ofrece el framework para el análisis de documentos.

UIMA inicialmente desarrollado por IBM, ahora bajo la administración de Apache, es un framework que permite generar sistemas para el análisis de grandes volúmenes de información no estructurada con el objetivo de encontrar información que sea relevante para un usuario final. Un ejemplo de una aplicación UIMA permitiría la lectura de un texto plano e identificaría entidades como personas, lugares, organizaciones o diversos tipos de relaciones.

UIMA permite y estimula a que las aplicaciones se descompongan en componentes, cada uno encargado de realizar un procesamiento específico en el documento analizado. Cada componente implementa un interfaz definida por el framework que provee la auto descripción de la metadata a través de la utilización de archivos XML. El framework administra estos componentes y el flujo de la información mediante su utilización. Los componentes son escritos en Java o C++, la información que fluye a través de éstos a sido diseñada para un mapeo eficiente entre estos dos lenguajes.

Además de los servicios antes mencionados UIMA brinda la posibilidad de publicar componentes como servicios web y poder escalar grandes cantidades de volúmenes a través de la replicación de los procesos sobre pipelines.

El uso del framework UIMA, incidió directamente en la arquitectura y separación en componentes. El desarrollo de múltiples componentes encargados cada uno de resolver una función específica y la información generada por cada uno, facilitaron la realización del proyecto.

5.2. Arquitectura de la solución

La arquitectura del sistema fue pensada para poder enfrentarnos a los principales desafíos y requerimientos del proyecto de la mejor forma posible.

Antes de comenzar la descripción, procederemos a enumerar los principales desafíos y requerimientos que fueron de influencia a la hora de diseñar la arquitectura.

5.2.1. Desafíos

- **Integración con Lavinia**: La solución al proyecto deberá poder ser utilizada desde Lavinia, lo que implica el desarrollo de componentes UIMA.

- **Análisis morfológico del documento**: Para poder analizar el texto, es necesario un análisis morfológico de éste.

- **Reconocimiento de relaciones**: Para poder realizar los algoritmos, es necesario contar con herramientas capaces de brindarnos información semántica del documento.

- **Algoritmo de identificación**: Existen múltiples algoritmos que pueden ser implementados con la información recopilada (morfológica, sintáctica, semántica), por esta razón se agregó como requerimiento, la posibilidad que el algoritmo de identificación pueda fácilmente ser intercambiado por otro.

5.2.2. Diseño de la arquitectura de la solución

Desde una perspectiva general (Figura 5.3), el sistema puede verse como una caja negra con dos entradas: una representando al documento a analizar y otra representando la expresión de consulta del usuario, y una salida representando al documento marcado con la solución.

Figura 5.3: Visión global

Como ya se ha mencionado, el proyecto requiere de la solución a varios problemas específicos e independientes. Esta situación radica en una ineludible división en componentes, como se puede ver en la Figura(5.4).

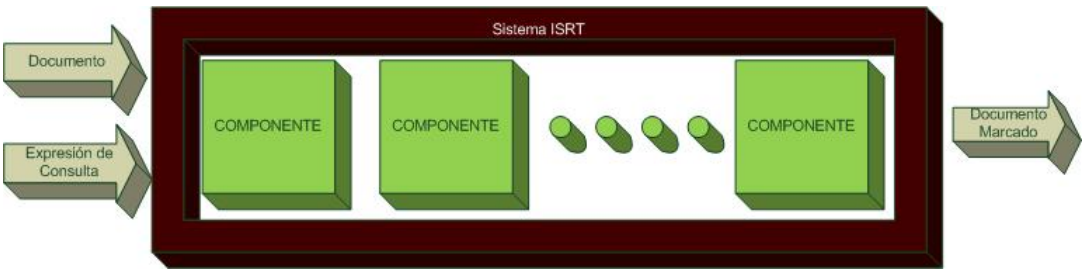


Figura 5.4: División en componentes

Si a la solución anterior, le agregamos los requerimientos de integración con Lavinia con la consecuente utilización de UIMA y los requerimientos propios de las herramientas a utilizar, llegamos a la arquitectura propuesta:

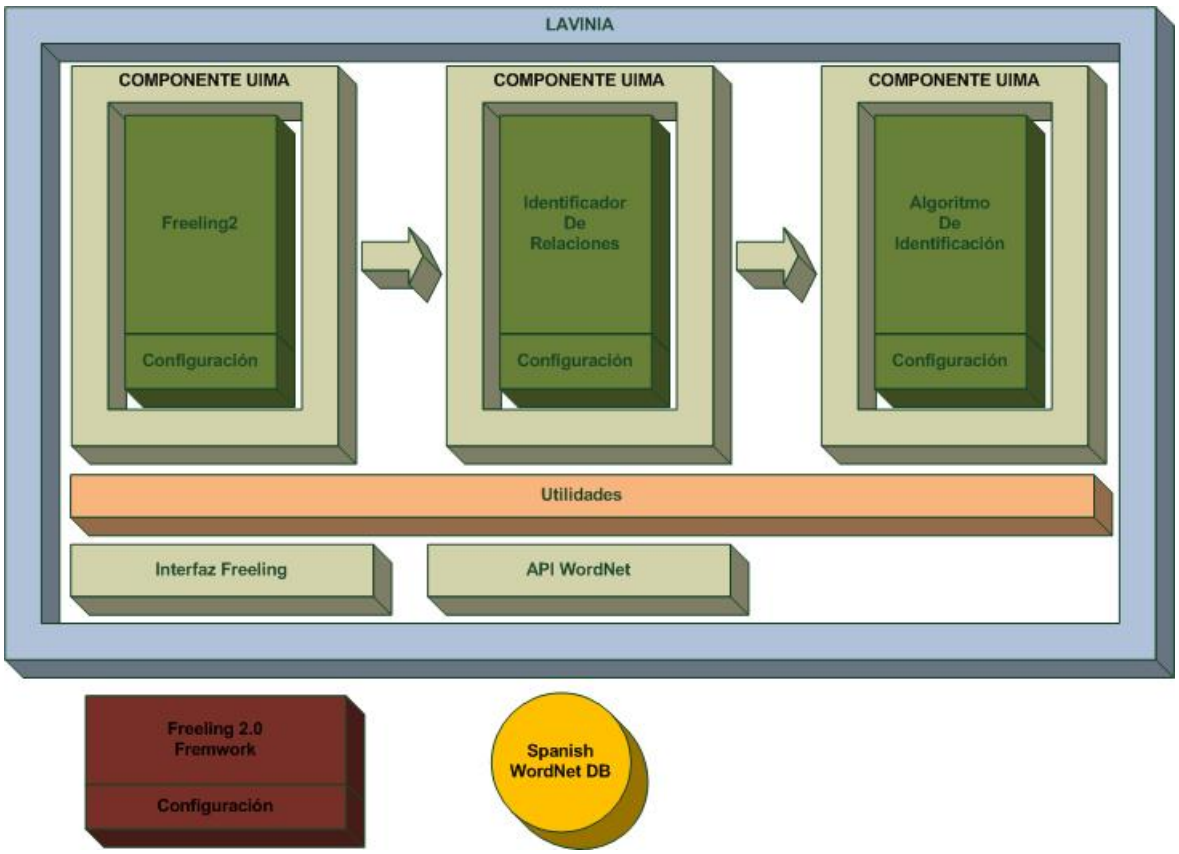


Figura 5.5: Arquitectura del Sistema

5.2.2.1. Descripción de los componentes

Para un entendimiento completo de la arquitectura propuesta, presentaremos una descripción de cada uno de ellos de forma individual.

Freeling2 - UIMA Component El componente *Freeling 2* (Tokenizador, Analizador Morfológico y Semántico), está encargado del análisis morfológico del documento, de la separación del documento en oraciones y también la separación de éstas en palabras o conjunto de palabras. Dentro de las principales funcionalidades que ofrece se encuentran:

- La tokenización del documento: palabras, palabras múltiples y oraciones.
- La identificación de entidades con nombre, locuciones, fechas, números, etc.
- La identificación de adjetivos, adverbios, verbos y otras categorías gramaticales.

Framework Freeling 2.0 El *Framework Freeling 2.0* es la herramienta externa que es utilizada desde el componente de Utilidades (el cual se presentará más adelante), para brindar los servicios ofrecidos por Freeling. Provee múltiples funcionalidades

para el análisis gramatical de documentos. Esta herramienta puede ser configurada para realizar diferentes tipos de análisis.

Identificador de Relaciones - UIMA Component El *identificador de relaciones* permite identificar en el documento todas las referencias directas e indirectas (por medio de distintos tipos de relaciones) de las palabras de la frase de consulta con las palabras que componen el documento.

Es un componente totalmente configurable, de forma tal que el usuario puede establecer qué tipo de relaciones quiere identificar, sus ponderaciones, ponderación referente a la profundidad para relaciones recursivas, entre otras configuraciones.

Para cumplir con las funcionalidades requeridas, este componente requiere la utilización de una herramienta externa que brinde las funcionalidades de diccionario semántico. Para esto se utilizó Spanish WordNet, que es una base de datos léxica para el español que contiene información sobre relaciones entre conceptos (*synsets*) y para cada concepto contiene un conjunto de sinónimos (*variants*) que lo representa.

API WordNet *API WordNet* es un componente que permite el acceso a la base de datos de Spanish WordNet. Se lo puede ver como una capa de acceso a datos (DataAccessLayer). Este módulo encapsula toda la complejidad de acceso a los datos mediante una serie de interfaces genéricas, así como también brinda un conjunto de data types definidos en el módulo para representar el resultado de las operaciones brindadas en las interfaces.

Es importante mencionar que se desarrolló en un módulo independiente para facilitar su uso posterior en proyectos y/o sistemas nuevos.

Spanish WordNet *Spanish WordNet* es esencialmente una base de datos relacional, administrada con MySQL. Contiene información sobre *synset*, *variants* y relaciones. Por más información ver Anexo A.2.3.

Algoritmo de Identificación - UIMA Component El *algoritmo de identificación* (identificación de segmentos relacionados), es el encargado de recuperar los resultados del análisis del documento. Utiliza toda la información recopilada por los demás componentes. Fue diseñado como un componente independiente para poder brindar y probar múltiples propuestas de soluciones.

Utilidades El componente de Utilidades, brinda una serie de herramientas comunes a los distintos componentes. Entre ellas accesos a recursos (por ejemplo Freeling), definición de constantes y estructuras de datos compartidas, métodos de ayuda para el uso de etiquetas EAGLES[5], etc.

[5] *EAGLES:* es un conjunto de etiquetas propuestas por el grupo EAGLES (*Expert Advisory Group on Language Engineering Standards*) para la anotación morfosintáctica de lexicones y corpus para todas las lenguas europeas.

Componentes UIMA Para cumplir el objetivo de integrar la solución con Lavinia creamos un contenedor UIMA al sistema que permita dicha integración.

Como consecuencia de la separación en componentes, podemos agregar a Lavinia cada uno de éstos fácilmente, agregando un contenedor UIMA a cada uno de ellos. Con esto se logra un valor agregado ya que cada componente podrá ser utilizado independientemente en Lavinia.

Lavinia Una vez desarrollados los componentes deberán ser integrados a Lavinia donde brindarán sus funcionalidades a los usuarios.

5.2.3. Comentarios sobre el diseño de la arquitectura

El diseño propuesto sigue los lineamientos propuestos por UIMA, para la especificación de la arquitectura de los sistemas de procesamiento de información no estructurada, separando cada parte específica del análisis de la información en componentes independientes unidos a través de flujos de información. La comunicación entre los componentes se realiza por interfaces definidas por el framework UIMA. Todos los elementos de la arquitectura son fácilmente sustituibles, permitiendo un fácil mantenimiento y perfeccionamiento del sistema.

5.2.4. Diseño

El diseño de la arquitectura del sistema desarrollado puede verse en el Anexo B, *Diseño*.

5.2.5. Puntos fuertes

La arquitectura propuesta presenta las siguientes ventajas:

- Fácil modificación de los componentes. Al utilizar UIMA, se especifican un conjunto de estructuras de datos (en un archivo de metadato XML de UIMA llamado *descriptor*) que se pasan o transfieren a través del flujo de los componentes. De esta forma se puede modificar la lógica interna de un componente de forma transparente a los demás.

- UIMA permite la utilización en forma individual y también componiendo un flujo, esto favorece el testing de cada componente.

- La configuración independiente de los algoritmos y los parámetros de entrada permiten su uso de forma versátil en la solución de múltiples problemas del análisis de documentos.

- La encapsulación de la capa de acceso a WordNet y la generación de un archivo JAR permite consultar la base de datos de WordNet desde cualquier componente o aplicación.

- El componente Framework Freeling 2.0 encapsula algunas de las funcionalidades presentes en Freeling. Mediante la publicación de interfaces y tipos de datos bien definidos hace transparente complejidades de como invocar al recurso y consumir sus resultados.

5.2.6. Puntos débiles y posibles mejoras

- Una posible mejora es permitir que en Lavinia los usuarios puedan interactuar con el sistema durante la ejecución de un flujo de componentes. Debido a está limitante no es posible la utilización del mecanismo de desambiguación manual en Lavinia.
- Los componentes necesitan de un "run-time environment" UIMA para funcionar.

5.2.7. Conclusiones sobre la arquitectura

Considerando los requerimientos del proyecto, apoyándonos en el estudio realizado y de las herramientas utilizadas, creemos que la arquitectura presentada como solución tiene todas las características necesarias para alcanzar el objetivo del proyecto.

5.3. Dificultades enfrentadas

En esta sección se presentan las principales dificultades encontradas durante el desarrollo del proyecto y cómo las mismas se resolvieron.

5.3.1. Configuración de Freeling

Freeling es una herramienta que brinda múltiples recursos lingüísticos para el análisis de documentos, su utilización fue imprescindible para el desarrollo del sistema. Actualmente Lavinia cuenta con un tokenizador y un postagger que hacen uso de Freeling. Una de las primeras decisiones que tuvimos que enfrentar fue si utilizábamos los componentes UIMA que brindaban servicios de Freeling ya existentes creados por el Grupo de PLN[6] del InCo[7] (en proyectos anteriores) o se construía un componente propio.

La decisión tomada fue esta última considerando los siguientes aspectos:

- Utilizar la última versión de Freeling porque esta provee nuevas funcionalidades y mejoras tanto en los algoritmos como en la base de datos.
- Tener control total sobre el parseo del documento y de esta forma realizar preprocesamiento que los componentes existentes no proveían como por ejemplo manejo de palabras múltiples, locuciones y determinación de las oraciones.

[6] http://www.fing.edu.uy/inco/grupos/pln
[7] http://www.fing.edu.uy/inco/pm/Main/HomePage

Parámetro	Valor	
InputFormat	plain	Texto de entrada plano
OutputFormat	tagged	Salida etiquetada
SenseAnnotation	all	Se incluyen todos los *offset*
SuffixAnalysis	yes	Se hace análisis de sufijos
MultiwordsDetection	yes	Se detectan palabras múltiples
NumbersDetection	yes	Se detectan números
PunctuationDetection	yes	Se detectan signos de puntuación
DictionarySearch	yes	Se hace búsqueda en diccionarios
ProbabilityAssignment	yes	Se agrega a los resultados la probabilidad

Cuadro 5.2: Parámetros de configuración de Freeling

Para poder crear nuestro propio componente Freeling, tuvimos que determinar de que manera utilizábamos Freeling como recurso. Las opciones fueron las siguientes:

- La utilización de una compilación en DLL para Windows mediante JNI[8].
- Realizar el desarrollo en C++ y utilizar directamente los paquetes de Freeling. Esto requiere un desarrollo para la plataforma Linux.
- Utilizar el ejecutable que provee Freeling para Windows.

La primer alternativa fue probada y funciona correctamente, la dificultad que encontramos fue al momento de la instalación de los componentes en Lavinia sobre todo la instalación de la DLL. La realización de todo el desarrollo en C++ no era posible ya que el framework UIMA necesario para la realización de los componentes está desarrollado para Java. Por último los componentes de Freeling que están presentes en Lavinia actualmente funcionan mediante la ejecución del programa analizador que provee Freeling. Este último punto fue el más fuerte para la elección del ejecutable provisto por Freeling para Windows como método de invocación.

Después de determinar la forma de invocación, fue necesaria la determinación de los parámetros de ejecución. El analizador que provee Freeling es capaz de aceptar un archivo de configuración en el cual se especifican dichos parámetros. En el Cuadro 5.2 se muestran algunos de los parámetros utilizados, los valores usados y el significado de estos.

5.3.2. Parseo de Freeling

El procesamiento del documento mediante Freeling tiene como salida un texto plano que debe ser analizado y asociado al documento que estamos analizando. Realizar este análisis resultó ser una gran dificultad debido a una serie de inconvenientes

[8] *Java Native Interface* (JNI) es la mecánica propuesta por Java para invocar funciones implementadas en lenguaje C desde programas Java.

que pasaremos a detallar junto con las posibles soluciones estudiadas y la solución finalmente implementada.

Lo primero para comprender la dificultad del procesamiento sería ver un ejemplo de cómo es una salida de Freeling. Para el texto *"El procesamiento de la salida de Freeling es más complicado de lo que parece."* Freeling genera la siguiente salida:

```
El el DAOMS0 1
procesamiento procesamiento NCMS000 1
de de SPS00 0.999919
la el DAOFS0 0.972146
salida salida NCFS000 0.8125 00029100:00031963:00032499:00201294
de de SPS00 0.999919
Freeling freeling NP00V00 1
es ser VSIP3S0 1 01666138:01775973:01784339:01787769:01817610
mas mas NCMS000 0.25
complicado complicar VMP00SM 1 00278110:00279361
de de SPS00 0.999919
lo el DAONS0 0.457393
que que PR0CN000 0.5625
parece parecer VMIP3S0 0.9875 00023357:01460069:01461214:01461433:01867887
. . Fp 1
```

Como se puede observar en el cuadro cada línea de la salida cumple con el siguiente formato:

<word> <lema> <EAGLES> <prob> [offset][:offsets]

Esta línea presenta las siguientes dificultades: primero la separación de cada componente de la salida; segundo la correcta lectura de cada uno de los *offsets* y por último la correcta lectura de los EAGLES. De estas dificultades, la más complicada es la lectura de los EAGLES ya que son de largo variable y cada letra que lo compone tiene un significado especial dependiendo de las anteriores.

Las dos primeras fueron solucionadas de forma sencilla mediante la partición de cada línea por el "espacio" para los distintos componentes y mediante la partición por ":" para los distintos *offsets*. Para la determinación de las propiedades morfológicas presentes en una etiqueta EAGLE se implementaron en el componente Utilidades un conjunto de clases (independientes de cualquier componente UIMA, u otro componente) que exponen un conjunto de métodos que dado una etiqueta EAGLE determina alguna propiedad (propiedad correspondiente al método, por ejemplo, en que tiempo se encuentra un verbo, o el tipo de un nombre(persona, lugar, etc.)) mediante las reglas definidas en la documentación de Freeling.

Una vez completado el procesamiento y obtenida la información del procesamiento de Freeling tenemos que asociar esta información a la palabra correspondiente en el documento analizado (es decir, determinar en que lugar en el documento se encuentran

las oraciones y las palabras identificadas). Esta asociación es casi directa, mientras vamos consumiendo el resultado del procesamiento de Freeling vamos consumiendo el documento y realizando la asignación; esto es posible debido a que el campo *<word>* que forma parte de la salida de Freeling "coincide" de cierta manera con la palabra que se está procesando.

El campo *<word>* correspondiente a la salida de Freeling no siempre contiene exactamente la palabra que se está procesando. Esto se debe a que Freeling es capaz de analizar y reconocer un conjunto de palabras múltiples y de más de 6.000 locuciones, por ejemplo para la frase *"Freeling funciona a las mil maravillas"* Freeling genera la siguiente salida:

```
Freeling Freeling NP00SP0 1
funciona funcionar VMIP3S0 0.75 01048914:01721982
a_las_mil_maravillas a_las_mil_maravillas RG 1
```

Como se puede observar la frase planteada presenta una expresión *"a las mil maravillas"*, y como es esperado, Freeling la reconoce como tal. Este comportamiento de Freeling es muy útil para el análisis de textos pero tiene como consecuencia una nueva dificultad en la asociación de los resultados de Freeling con el documento analizado. A simple vista se ve que el valor del campo *<word>* resultante para la locución no corresponde directamente con ninguna palabra de la frase analizada, esto se debe a que en Freeling todas las locuciones o palabras múltiples son separadas por un guión bajo. La solución implementada fue la de buscar cada palabra de forma independiente en el documento, es decir, tratar de encontrar la frase *"a las mil maravillas"* de forma que puedan existir cualquier separador entre las palabras (espacios, tabuladores, retorno de carro y salto de línea, etc.). Luego de identificar todas las palabras, se anota toda la frase como una unidad en conjunto, esto es desde el comienzo de la primer palabra hasta el final de la última.

Otra dificultad que se suma es el hecho que Freeling no reconoce las contracciones como tal sino que las divide en sus componentes. Por ejemplo, la palabra *"del"* la descompone en dos (*"de él"*) resultados no como una palabra compuesta sino como dos líneas diferentes en el archivo de salida, esto mismo sucede con la palabra *"al"* que Freeling descompone en *"a él"*. Como es lógico cuando estamos procesando el documento no tenemos dos palabras para etiquetar, sólo tenemos una, por tal motivo asignamos las dos palabras a la palabra *del* o *al* del documento según corresponda.

Consideramos que esta no es la mejor solución, pero la elegimos frente a otra por ser mucho más eficiente. La alternativa sería para cada palabra próxima a asociar del documento, procesarla primero con Freeling. De esta forma, podríamos mapear directo las palabras con el resultado. Esta opción se rechazó debido que el procesamiento de Freeling es lo que ocasiona la mayor demora en la ejecución de los algoritmos y esta solución requiere de múltiples invocaciones. Además Freeling trabaja con la totalidad del texto, es decir utiliza el contexto de las palabras dentro de la oración, por ejemplo para determinar la categoría gramatical.

Existen muchas locuciones en español que tienen contracciones dentro de su conjunto de palabras; estos casos se solucionaron de forma similar a cuando las contracciones

se dan fuera de locuciones.

Ejemplo de locuciones que contienen contracciones son: *"administración del estado"* Freeling devuelve como *<word> "administración_de_el_estado"*. *"al cierre del ejercicio"*, *"al fin y al cabo"*, *"al hilo del viento"* son otras locuciones de Freeling que poseen contracciones. El parseo del resultado de Freeling y la asociación de estos con cada palabra del documento, es sin duda la piedra angular de todo el sistema; el correcto funcionamiento es imprescindible. Con respecto a la solución propuesta cumple satisfactoriamente con los requerimientos y no tiene la contra de la realización de múltiples invocaciones a Freeling. La solución implementada fue testeada con el conjunto completo de las locuciones reconocibles por Freeling y con un conjunto de más de 50 documentos.

5.3.3. Selección del mejor offset

Para encontrar en el documento aquellas sentencias que están vinculadas a la frase de consulta, tenemos primero que procesar la consulta. El primer procesamiento que le realizamos y punto de partida para posteriores análisis es el procesamiento por Freeling.

Dicho procesamiento nos provee mucha información morfológica sobre la expresión de consulta y además se obtiene información de la colección de *offset* asociados a cada palabra. Los *offsets* que forman parte del resultado del procesamiento para una palabra corresponden a los posibles significados de la misma (por ejemplo la palabra *"auto"* tiene como *offsets "05264886, 04909380, 02383458"* donde cada uno corresponde a un significado diferente de la palabra). El problema al que nos enfrentamos es poder identificar cual de los *offset* corresponde al significado deseado por el usuario en la consulta. Este no es un problema trivial y no existe un método capaz de resolverlo automáticamente. Este problema se conoce como *"Desambiguación del Significado"* (*Word Sense Disambiguation*). La selección incorrecta del significado de una palabra tiene como consecuencia que se pierdan muchos resultados posibles y peor aún que se encuentren resultados que son incorrectos. No es raro que el propio ser humano tenga dificultad para discernir el significado de una palabra para un contexto, lo que hace razonable pensar lo dificultoso que es que el reconocimiento se realice de forma automática. Aún considerando esto, se han realizado algunos trabajos enfocados en resolver este problema.

Se analizaron las siguientes soluciones:

- La primera hace uso de diccionarios restringidos para un dominio específico, lo cual permite que cada palabra tenga un significado único. Este método tiene la contra que el texto y la frase de consulta tienen que pertenecer al mismo dominio y que el usuario debe conocer y entender el dominio. Por lo anterior este método no pudo ser implementado por nuestro sistema ya que no está acotado a un dominio específico.

- El segundo enfoque intenta desambiguar el significado a través del contexto en que se encuentra la palabra. Este método es más complejo que el primero y

tiene la ventaja de no estar atado a un dominio específico. Una de las debilidades que tiene es que como la frase del usuario es pequeña puede no proporcionar suficiente información de contexto como para poder desambiguar correctamente.

- La tercera opción consiste en que la elección del significado sea realizada por el propio usuario, esta es la solución que ofrece mejores resultados pero requiere la intervención directa de una persona (un ejemplo de esta solución es la implementada por Wikipedia[9]). En muchos casos esta interacción es imposible o dificultosa. Por ejemplo en la implementación actual de Lavinia, no es posible la interacción con el usuario cuando el framework se encuentra procesando el flujo.

- La última solución analizada es utilizar el significado que se usa con más frecuencia. Esta opción muestra una correcta desambiguación en la mayoría de los casos y aumenta su eficacia con respecto a las anteriores propuestas cuanto más general y variados sean los documentos y las personas que realizan las consultas. Esto se debe a que en general los usuarios eligen las palabras de las consultas pensando en el significado más frecuente. Para hacer uso de esta solución necesitamos de un diccionario que mantenga para cada palabra no solamente los posibles significados sino que también tenga el más común.

Una característica interesante del sistema implementado es que permite parametrizar la solución a utilizar. Las posibles opciones que el sistema brinda son:

- *Utilizar todos los offsets.*

- *La utilización del offset (significado) más frecuente.*

- *Permitir que el usuario, para cada palabra relevante de la frase de consulta, determine cual es su significado (de esta forma determine el offset).*

Es importante destacar que para realizar esta parametrización no es necesario recompilar el proyecto. El sistema se diseñó de tal forma que se puedan modificar los parámetros de ejecución; quedando así un sistema más usable y flexible. Esta decisión de diseño también la llevamos a otras propiedades del sistema y fue de gran ayuda para poder probarlo, sacar conclusiones de sus resultados con diferentes parámetros y poder elegir la mejor configuración que nos llevara a mejores resultados.

En conclusión, consideramos que la solución propuesta para este problema es satisfactoria en la mayoría de los casos y se deja planteado como trabajo futuro la implementación de una solución mejor, como por ejemplo la desambiguación con intervención del usuario en Lavinia o la desambiguación del significado de forma automática aliándose de técnicas estadísticas de aprendizaje automático.

[9] http://es.wikipedia.org/

5.3.4. Obtención de Spanish WordNet

WordNet es una enorme base de datos léxica que provee información muy importante para el desarrollo del proyecto. Sin embargo para poder usar WordNet es necesario la adquisición de una licencia que permita su uso. Para obtener la licencia buscamos la información necesaria para el trámite, y gracias a los docentes del proyecto, pudimos enviar una carta sellada por la facultad solicitando una licencia gratuita con fines académicos. Después de algunas semanas de espera obtuvimos la respuesta esperada y nos fueron entregados varios scripts que nos permitieron generar la base de datos.

Sin duda la obtención de WordNet fue un punto clave en el desarrollo del proyecto y un resultado extra fue que la Facultad de Ingeniería pudo obtener una licencia y la base de datos de una importante herramienta para el análisis léxico de documentos que sin duda será utilizada no sólo en futuros proyectos sino también en cursos existentes.

5.3.5. Integración con Lavinia

Si bien nuestros componentes implementan el framework UIMA, la instalación de los componentes en Lavinia no fue una tarea sencilla. Los principales problemas encontrados en la integración fueron: no todos los tipos soportados por UIMA son soportados en Lavinia, por ejemplo la información presente en un *annotator*[10] no pueden contener listas, para subsanar esto se utilizaron Arrays. Otro más importante aún es que Lavinia no acepta la separación de los *descriptors* de los componentes. Esta separación es muy importante al momento del desarrollo para no tener que definir en cada componente los tipos y anotaciones que son de entrada de componentes anteriores en el flujo. De esta forma se podría hacer referencia a la definición de los mismos.

El problema de los tipos fue solucionado no utilizando los tipos conflictivos, sin embargo consideramos que sería pertinente la actualización de Lavinia para que maneje esos tipos correctamente. El problema de los descriptors pudo resolverse uniendo los descriptores de los componentes antes de realizar la integración.

5.3.6. Dominio no acotado

El hecho de permitir el procesamiento de documentos de carácter general (es decir de cualquier dominio, formato y propósito) sumado a que los usuarios del sistema también lo sean, imposibilita la posibilidad de determinar un dominio acotado. Este factor tiene como consecuencia directa la imposibilidad de utilizar ontologías específicas y diccionarios especializados.

[10] *Annotator*: Estructura de datis utilizada por UIMA que permite adjuntar información procesada al documento analizado.

5.3.7. No existencia de herramientas semejantes

Por último, la principal dificultad enfrentada es sin duda el hecho de no haber encontrado ninguna herramienta similar disponible para ser analizada. Esto ocasionó que toda la implementación e investigación fuera sustancialmente más complicada e intensa al no tener un modelo guía para seguir, ni para comparar los resultados.

5.4. Implementación

5.4.1. Decisiones tomadas

Durante el desarrollo de la solución propuesta, se presentaron un conjunto de problemas y decisiones de implementación que tuvimos que analizar con el objetivo de encontrar la mejor solución posible. Es en esta sección se presentarán algunas de las decisiones más importantes.

Separar los analizadores en componentes

En un momento de la implementación, y apoyados por el diseño de la arquitectura, se analizó la posibilidad de que los analizadores de relaciones que forman parte del componente *analizador de relaciones* fueran componentes independientes. Esto fue descartado por las siguientes razones:

- La cantidad de componentes se incrementaría, esto tiene como consecuencia un aumento en la complejidad del sistema, principalmente dificultades de configuración y creación de flujos.
- La ventaja principal de la división es la de permitir al usuario (principalmente de Lavinia) la utilización de los componentes individuales y unirlos a otros flujos. Para evitar perder ésta ventaja, se diseñó el componente de forma que el usuario pueda establecer parámetros de invocación de forma que el componente analice únicamente aquellas relaciones deseadas por el usuario.
    - Por ejemplo, un usuario podría necesitar encontrar todos los hiperónimos de un texto para después procesarlos con un componente propio. Si este es el caso el usuario podría utilizar el componente *Freeling2* para procesar el documento, pasarle el resultado al componente de *Identificación de Relaciones* (configurándolo de forma que solo identifique los hiperónimos) y utilizar la salida como entrada en el componente del usuario.
- La agrupación de los analizadores en un componente genérico permite la utilización de herramientas genéricas que disminuyen notablemente el código y aumentan la eficiencia, tanto para el desarrollo como en la ejecución.
- Reduce considerablemente la cantidad de archivos de descripción UIMA, necesarios para la configuración del sistema.

Extender la frase de consulta

Encontrar la relación entre palabras que forman parte de la consulta y las que forman parte del documento es clave para cumplir con el objetivo de nuestro proyecto. Sin embargo la relación directa entre dos palabras sólo se produce cuando esas palabras son idénticas o si tienen el mismo lema si se posee esta información. Más difícil aún es lograr establecer relaciones entre una palabra y sus sinónimos, y más aún, lograr establecer una relación entre palabras relacionadas por una relación de meronimia (relación que puede tener múltiples niveles de profundidad).

Para poder lograr determinar este tipo de relaciones, es necesario poder asociar a cada palabra un conjunto mayor de posibles candidatos (por ejemplo, sinónimos, hiperónimos, hipónimos, etc.). A esto lo denominamos *"extensión de la palabra"*. A la extensión del conjunto de las palabras de la frase brindada por el usuario del sistema denominamos *"extensión de la frase de consulta"*.

En otras palabras, extender la frase de consulta implica aumentar de una forma inteligente la cantidad de palabras que existen en ella, con el objetivo de encontrar un mayor número de relaciones.

No extender el documento completo

Como vimos en el punto anterior la extensión de la frase de consulta y del documento es clave para encontrar un mayor número de relaciones, sin embargo realizar dicha expansión es una tarea computacionalmente costosa. La expansión de la consulta es posible dado que la cantidad de palabras que forman parte de una consulta es un número reducido y acotado. En cambio, el documento completo es por lo general mucho más grande y la cantidad de palabras que lo conforman es variable. Esta es la principal razón de no haber expandido las palabras del documento completo.

Una de las consecuencias que tiene no extender el documento es que se perderán algunas relaciones. Para minimizar esta desventaja se utilizaron los mecanismos de: aumentar el número de tipo de relaciones analizados y principalmente analizar las relaciones inversas.

Por ejemplo, *"es parte de"* es una relación directa, ahora si la expresión de consulta es *"moto"* y en documento aparece la palabra *"rueda"*, estas no podrían ser relacionadas debido a que la *"moto"* no es parte de la *"rueda"*. En cambio si utilizamos la relación inversa si podemos relacionar los conceptos.

Otra consecuencia es que imposibilita la creación de un algoritmo para identificar el tema de un documento, ya que este analizador necesitaría que el documento estuviera procesado completamente.

Se consideró la posibilidad de crear un componente independiente que permite el procesamiento completo del algoritmo, pensando en brindar una funcionalidad extra a Lavinia, pero se desestimó esta opción por falta de tiempo para su desarrollo. Pero sí se presenta como una posible mejora al sistema desarrollado.

Lista de palabras ignoradas

En el componente *Identificador de Relaciones*, más precisamente en la determinación de los lemas que coincidan entre las palabras de la frase y las del documento, se considera que la identificación de ciertas palabras no aporta información relevante sobre si la oración que la contiene corresponde con la frase de consulta, sino que por lo contrario, agrega ruido a los resultados. Ejemplo de estas palabras son, las palabras gramaticales (o *stopwords*, ej: "de", "la", "y", "o"), los signos de puntuación, las palabras con las siguientes categorías gramaticales: los determinantes, las conjunciones, las interjecciones, las preposiciones, los pronombres, etc. Un ejemplo de determinar los lemas coincidentes sin filtrar los términos anteriormente mencionados es el siguiente: dada la frase *"El auto de mi hermano es rojo"* y las oraciones *"El color rojo es usado en el semáforo"* y *"Mi hermano tiene un auto rojo"*, se podría decir a simple vista que la primer oración tiene un valor de 4 y la segunda un valor de 2, sin embargo la oración que contiene la información consultada es la segunda. El problema radica en que hay un conjunto de palabras que no aportan información semántica importante y que además son usadas con mucha frecuencia. Por este motivo se planteó la necesidad de la creación de una "lista de palabras" que no serán consideradas para la valoración de las oraciones, para esto se implementó un servicio en el componente Utilidades que dado una etiqueta EAGLES determina si tiene sentido comparar los lemas (coincidentes) asociados a la etiqueta.

Relaciones

Existe una gran variedad de relaciones entre las palabras del idioma español. Lo ideal sería poder analizar todos los tipos de relaciones pero por diferentes motivos esto no es posible.

La primera limitante del conjunto de relaciones a analizar es que no existe ninguna herramienta que mantenga catalogadas el conjunto completo de relaciones posibles. Sí existen herramientas que proveen un conjunto importante (por ejemplo WordNet) de las mismas.

Otra limitante es generada por la poca cantidad de conceptos que se encuentran relacionados por una determinada relación. Por ejemplo *Spanish WordNet* para la relación HAS_MERO_PART existen únicamente 28 instancias (registros), en cambio su relación "inversa" HAS_HOLO_PART posee 6881 registros. Por esta razón se decidió identificar la inversa de HAS_HOLO_PART como si se tratara de HAS_MERO_PART (esto se ha realizado también para otras relaciones). Esto lleva a poder identificar un mayor número de relaciones pero con un mayor margen de error. Un ejemplo de esto sería: sabemos que un *auto* tiene *puertas*, pero una *puerta* puede pertenecer a un *auto*, una *casa*, un *ómnibus*, etc.

Las relaciones a utilizar para la identificación se pueden configurar. De esta manera, es posible habilitar y deshabilitar relaciones como las del ejemplo anterior. Por ejemplo cuando existan más conceptos relacionados por la relación HAS_MERO_PART se podría deshabilitar el uso de la "inversa" de HAS_HOLO_PART.

Para el análisis de relaciones fue necesario especificar una serie de parámetros. Entre ellos se encuentran: la ponderación de la relación, profundidad del reconocimiento recursivo, y el reconocimiento o no de su inversa. En el Cuadro 5.3 se presentan los parámetros utilizados.

Relación	Ponderación	Recursiva	Inversa
lema	15	no	no
sinónimo	15	no	no
near_synonym	10	no	no
has_hyponym	10	si	si
has_holo_member	6	si	si
near_antonym	5	no	no
has_holo_part	4	si	si
has_derived	4	no	si
has_xpos_hyponym	2	no	si
has_holo_madeof	2	si	si
xpos_near_synonym	2	no	si
role_agent	1	no	si
has_mero_member	1	si	si
has_mero_part	1	si	si
has_subevent	1	no	si
pertains_to	1	no	no
role	1	no	no
be_in_state	1	no	si
causes	1	no	si

Cuadro 5.3: Configuración de las relaciones

Utilizar las inversas de las relaciones de WordNet

Si bien WordNet provee numerosas relaciones, no todas están presentes implícitamente. De todas formas, algunas de las que no están presentes igualmente pueden ser inferidas. Ejemplo de esto son los hipónimos e hiperónimos, si bien los hiperónimos no existen en WordNet si existen los hipónimos y como la relación *"tiene hiperónimo"* es la inversa a la relación *"tiene hipónimo"* podemos encontrar estás realizando la consulta inversa.

La decisión de analizar las relaciones inversas fue tomada por las siguientes razones: la primera es que las relaciones inversas son tan usadas y proveen distinta información que las relaciones directas; la segunda es que el análisis de las relaciones inversas permite mitigar una de las desventajas del hecho de no extender el documento completamente.

Criterio para la selección de los resultados

En el sistema existen 3 algoritmos de selección de resultados que, basados en la información recompilada en los componentes anteriores, determinan cuáles son las oraciones que están relacionadas con la expresión de consulta.

A continuación se presentan algunas opciones analizadas:

- En primera instancia se podría considerar reconocer como resultado todas aquellas oraciones que tengan un valor positivo. Esto implica que toda oración donde se haya podido identificar algún tipo de relación con la consulta será considerada como resultado. La consecuencia directa de esta elección es que se estarían devolviendo muchos resultados que aunque tienen alguna palabra relacionada con la consulta, ésta realmente no trata los temas de la consulta. Esto ocasiona que se incremente el valor de *recall*[11] al devolver más resultados, pero que se baje drásticamente la *precision*[12].

Veamos lo que sucede para un conjunto de diez oraciones cuyos valores son 0, 0, 1, 9, 5, 1, 3, 0, 0, 0.

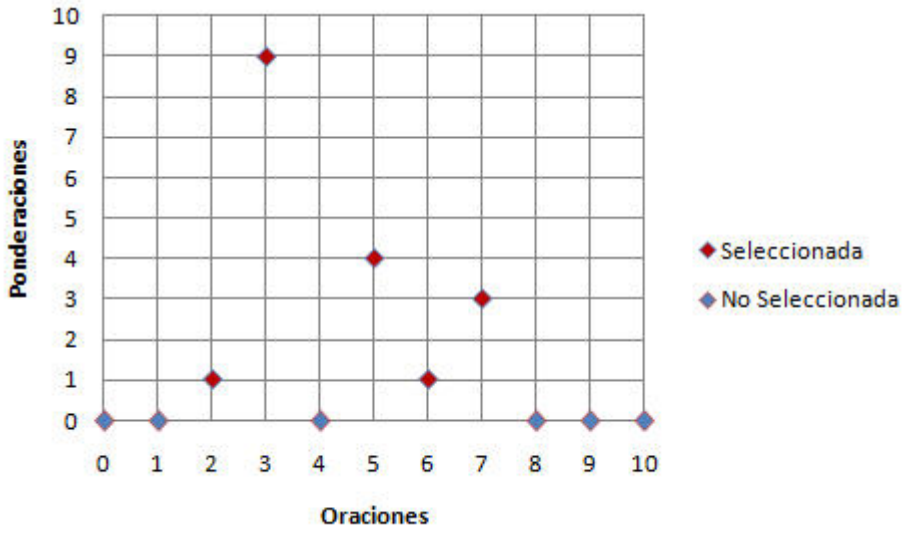


Figura 5.6: Oraciones seleccionadas

En la Figura 5.6, se puede observar como se estarían retornando las oraciones 2 y 6, que tienen poca relación respecto a la consulta.

Una variante a este algoritmo es ordenar las oraciones de forma descendente según su ponderación y devolver las K primeras. Obviamente (dependiendo del valor de K) ésta solución tiene un menor *recall* y una mayor *precision*, pero puede ocasionar que oraciones que realmente son relevantes para el usuario no sean identificadas.

[11] *Recall*: Cantidad de oraciones correctamente identificadas sobre el total de oraciones que debían ser identificadas.

[12] *Precision*: Cantidad de oraciones correctamente identificadas sobre el total de oraciones identificadas.

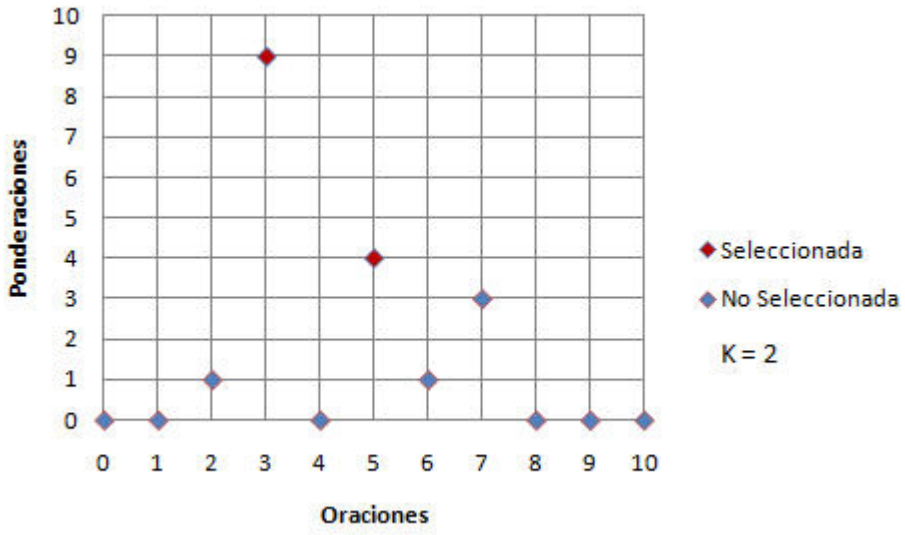


Figura 5.7: Oraciones seleccionadas, para K = 2

Como podemos ver en la Figura 5.7, la oración 7 que tiene una ponderaración importante no es retornada. Con este método también puede ocurrir que para un valor K grande se retornen como solución oraciones poco relevantes como la 2 y 6. Este criterio de selección de aquí en más lo denominaremos *"Algoritmo de selección 1: K mejores"*.

- La segunda opción analizada fue la de asignar a la frase de consulta del usuario un valor (a este valor denominaremos *"valor frase de consulta"*, VFC) que está determinado por la cantidad de palabras de la consulta multiplicado por el valor de la ponderación para los lemas, lo que correspondería a identificar en el documento la misma frase de consulta o una muy similar.

$$VFC = CantPalabras(expresión\ de\ consulta)\ *\ VALOR\_LEMA$$

El VFC servirá para decidir si una oración es lo suficientemente buena para ser considerada como resultado. De esta forma todas las oraciones del documento que cumplan que su valor es mayor o igual a una constante K por el *"valor frase de consulta"*, serán consideradas como buenas aproximaciones y añadida como solución posible. A esta solución la denominaremos *"Algoritmo de selección 2: aproximación por VFC"*.

La oración O es posible respuesta si:

$$Valor(O) \geq K.VFC$$

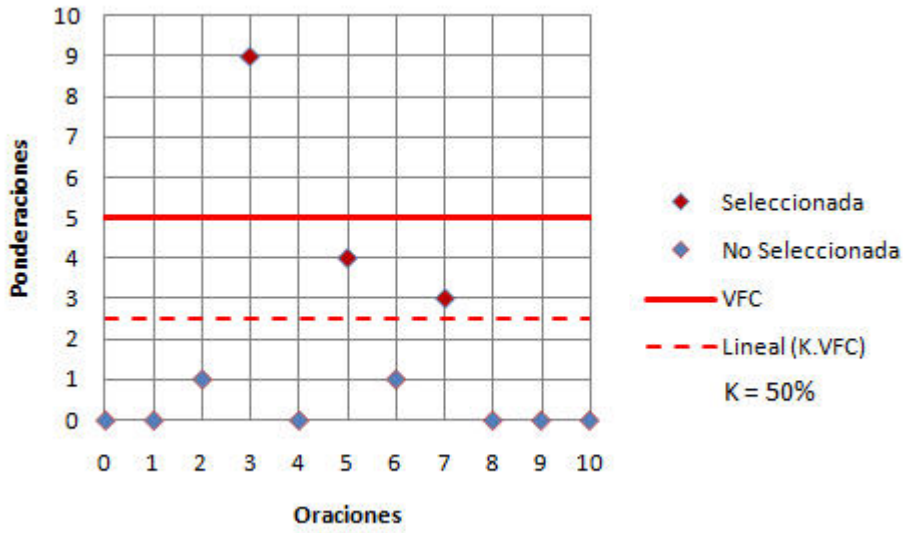


Figura 5.8: Oraciones seleccionadas, Algoritmo de selección 2, K = 0,5

Mediante este algoritmo y volviendo al ejemplo, nos aseguramos que las oraciones reconocidas estén relacionadas con la consulta del usuario. Una consulta con muchas palabras tendrá un valor alto, por lo que solo oraciones con múltiples relaciones serán consideradas. Por otra parte todas aquellas que tengan pocas relaciones, si bien tienen algo en común, son descartadas al no asemejarse lo suficiente con la consulta del usuario. Este método también funciona cuando la frase de consulta es corta e incluso de una única palabra, esto se debe a que el VFC en este caso sería pequeño y permitiría reconocer oraciones en las cuales aparece solo una relación con la palabra ingresada como consulta.

- El último criterio analizado utiliza conceptos de valor esperado (VE), varianza y desviación estándar. Para todas las oraciones con ponderación positiva, encuentra el VE o esperanza (en este caso tiene igual valor que el valor promedio). Luego se calcula la varianza mediante la sumatoria[13]:

$$varianza = \sum \frac{(ponderación\ oración - VE)^2}{cantidad\ oraciones\ con\ ponderación\ positiva}$$

Por último se calcula la *desviación estándar* mediante el cálculo de la raíz cuadrada de la varianza y se seleccionan todas las oraciones que tengan una ponderación mayor o igual a: *VE - desviación estándar*, es decir, esta es la cota mínima admitida. Este criterio de selección lo nombramos *"Algoritmo de selección 3: valor promedio"*.

Para este algoritmo de selección, la oración O es posible respuesta si:

$$Valor(O) \geq VE\ -\ desviación\ estándar$$

[13]La sumatoria aplica para cada oración con ponderación positiva.

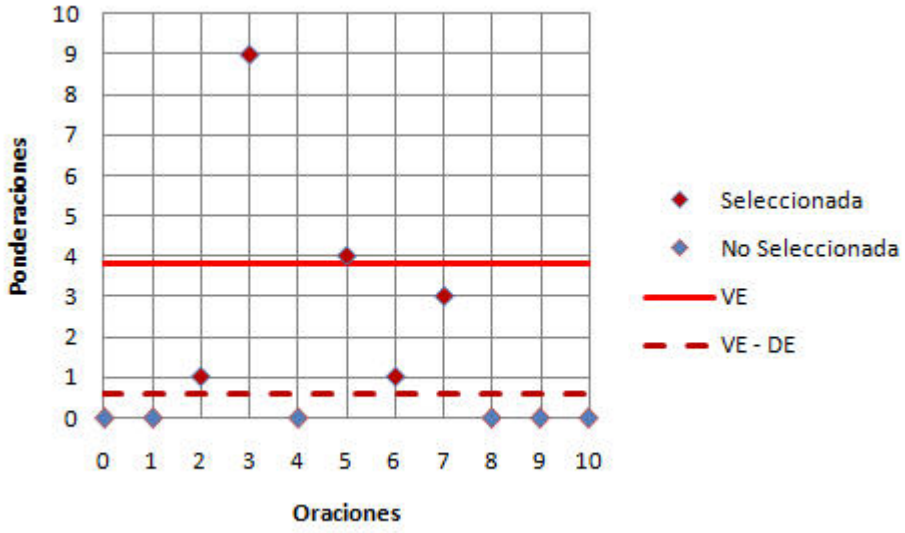


Figura 5.9: Oraciones seleccionadas, Algoritmo de selección 3

La línea punteada en la Figura 5.9 representa la cota mínima de selección obtenida a partir del valor esperado y la desviación estándar. Los resultados de este algoritmo están fuertemente asociados a que tan apartados se encuentran los valoresde las ponderaciones, cuanto mayor sea esta dispersión menor será la cota inferior, permitiendo una mayor selección de resultados.

5.4.2. Detalles de implementación

Ventana modal para desambiguar el significado de la frase de consulta

Con el objetivo de resolver el problema de la desambiguación semántica con apoyo del usuario (por más información ver la subsección 5.3.3), se desarrolló una ventana que aparece en el momento del procesamiento de la consulta del usuario. La idea es que el usuario pueda interactuar con el sistema, seleccionando para cada palabra que forma parte de la consulta el significado que mejor se adapta de acuerdo a su parecer. Es preciso que la interfaz brindada sea amigable e intuitiva y que brinde una preselección de los significados más comunes para cada palabra de la consulta.

La ventana del desambiguador fue realizada en Java y el utilizando las librerías que provee el framework Java Swing las cuales brindan funcionalidades para el desarrollo de aplicaciones de escritorio.

El sistema separa la frase de consulta *"Esta es la ventana de determinación de el significado"* en un conjunto de palabras, ver Figura 5.10. Para cada una de las palabras se obtienen el conjunto de *synsets* asociados, donde cada uno de estos *synsets* corresponde con un significado posible de la palabra analizada. Para asistir al usuario al momento de seleccionar el significado deseado de cada una de las palabras se agregó la información de la definición del *synset* (si existe en WordNet) y el conjunto de sus

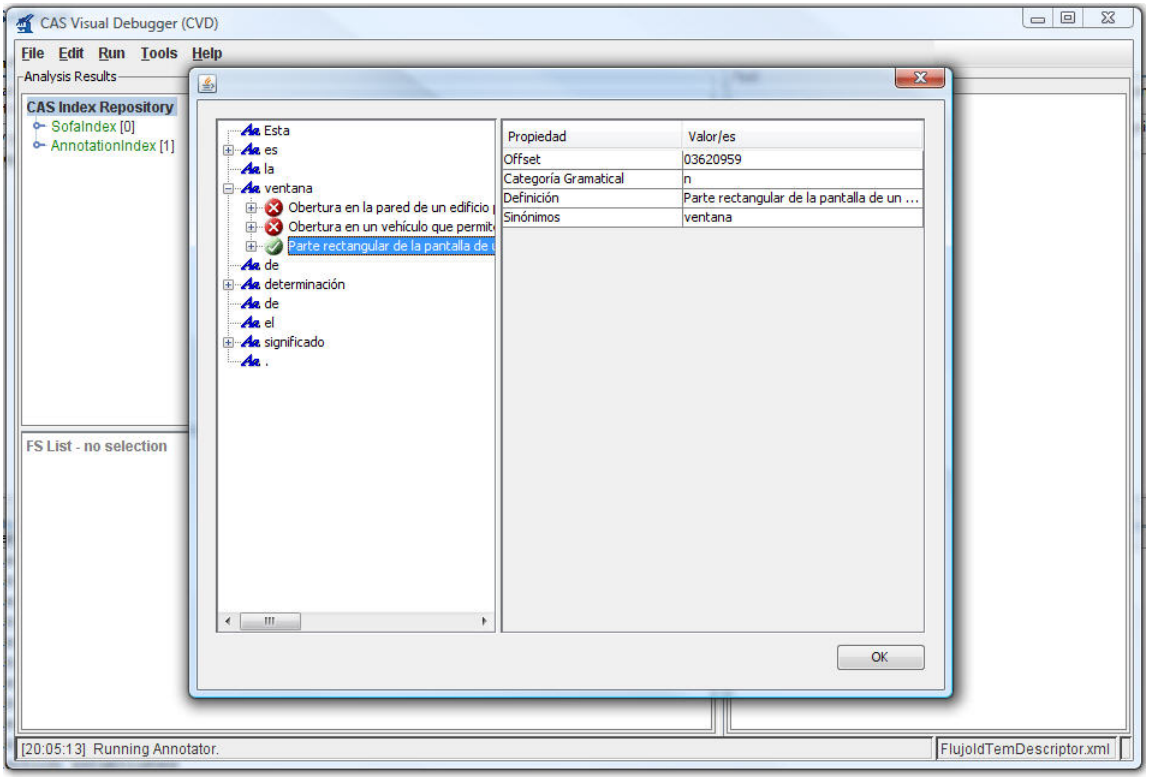


Figura 5.10: Interfaz gráfica ventana modal

hiperónimos e hipónimos como se ve en la Figura (5.10). Estos fueron incluidos debido a que en WordNet menos del 10 % de los *synsets* tienen especificado su definición. De esta forma, para los casos que no se tiene la definición, se brinda un contexto al usuario mostrando los hipónimos y los hiperónimos de un *synset*.

Esta solución no pudo ser utilizada desde Lavinia, dado que es una aplicación web y no es posible abrir una ventana windows. Como consecuencia de esto, se agregó un parámetro de configuración que permite indicar si se utilizará o no la ventana de desambiguación. Se dejará como posible extensión de Lavinia la implementación de algún mecanismo que permita realizar tareas semejantes a la anterior, la cual requiere la intervención del usuario en algún instante del procesamiento de un flujo de componentes UIMA.

6 Pruebas realizadas

Nuestro sistema no es un sistema de extracción de información convencional, si bien posee muchas características propias de estos sistemas también posee características de los sistemas de recuperación de información, principalmente en lo referente al ingreso de una expresión de consulta. Estas diferencias hacen imposible la utilización de los métodos de evaluación propios de sistemas de EI convencionales (ver Sección 3.4).

Para salvar estas diferencias fue necesario la utilización de una metodología de testing combinada, tomando rasgos de los métodos de pruebas para sistemas de RI y EI. A continuación describiremos el proceso de pruebas utilizado.

6.1. Descripción del proceso de pruebas

- **Definición de los datos de prueba**: Esta etapa consiste en especificar las características de los distintos elementos (por ejemplo: parámetros de configuración, documentos de entrada y frases de consulta) que se utilizarán en las pruebas del sistema y sus relaciones.

- **Determinación de las frases de consulta y documentos para las pruebas**: La determinación de un buen conjunto de frases y documentos es imprescindible para una correcta validación. Estos deberán permitir que el sistema se enfrente a las principales dificultades propias del problema. Dentro del conjunto de frases de consulta deberá haber frases identificables, no identificables, con problemas de correferencia, anáforas, desambiguación, palabras sueltas para la identificación de diferentes tipos de relaciones como hiperónimos, hipónimos, etc. Los documentos deberán ser lo suficientemente grandes como para poder realizar las identificaciones, pero no demasiado, ya que ocasionaría una demora importante en las pruebas. Se pretende que los documentos que forman parte del corpus de pruebas tengan entre 1 y 5 páginas.

- **Análisis manual del corpus**: Se deberá procesar el corpus de documentos manualmente marcando las posibles identificaciones para cada frase de consulta. El objetivo es poder comparar los resultados de la identificación automática con la identificación manual realizada por una persona.

- **Definición de los parámetros del sistema**: Antes de comenzar con las pruebas automáticas es necesario establecer la configuración de distintos parámetros del sistema.

- **Análisis automático del corpus - Desambiguación Automática**: Utilizando el sistema desarrollado se deberán procesar los documentos y las frases de consulta, utilizando un método automático para la desambiguación de los significados.

- **Análisis automático del corpus - Desambiguación Manual**: Utilizando el sistema desarrollado se deberán procesar los documentos y las frases de consulta, utilizando un método manual para la desambiguación de los significados.

- **Comparar los resultados manuales y los automáticos**: Se deberán comparar los resultados de la identificación manual y la automática, explicando el motivo de las divergencias que se produzcan. Realizar cálculos de *precision, recall* y *F-measure*.

- **Conclusiones**: Dar conclusiones de las pruebas y cálculos realizados**.**

6.1.1. Definición de los datos de prueba

Los elementos variables del sistema son: los parámetros de configuración, los documentos de entrada y las frases de consulta. Para las pruebas se eligió una configuración como base y el único parámetro variable seleccionado corresponde con el tipo de desambiguación, por lo que los principales elementos que se utilizarán para ver la eficacia del sistema son el conjunto de documentos y el conjunto de frases de consulta.

En este punto es importante lograr establecer los criterios para la selección de las frases y documentos que se usarán, buscando que los conjuntos sean lo más pequeños posibles para facilitar las pruebas, pero sin perder confiabilidad, rigurosidad y generalidad.

Los criterios establecidos para la selección de textos son los siguientes:

- Se generarán 5 conjuntos (G1, G2, G3, G4, G5) de documentos, cada uno formado por 3 documentos, los cuales estarán relacionados temáticamente con un conjunto de 4 frases de consultas. La idea es que cada uno de estos conjuntos trate los temas de la frase de consulta y que éstas sean identificables en los documentos.

- Se generará 1 conjunto (GDNR) de 3 documentos tomados al azar. Para este conjunto se probarán todas las frases de los conjuntos anteriores y servirá para analizar cómo se comporta el sistema cuando la frase de consulta no debe ser identificada en los documentos.

- Consultas suficientemente genéricas que permitan probar múltiples problemas propios de la identificación. Además que permitan identificar en los documentos distintos tipos de relaciones: sinónimos, hiperónimos, hipónimos, etc.

6.1.2. Determinación de las frases de consulta y documentos de pruebas

Para cumplir con las especificaciones en los datos de prueba, tuvimos que construir el conjunto de documentos y de frases al mismo tiempo.

La idea principal es dividir el corpus en 6 subconjuntos, los primero 5 estarán relacionados con las frases de consultas y el último estará formado por documentos no relacionados.

Para la obtención de los documentos relacionados seguimos los siguientes pasos:

1. Se seleccionó un tema genérico, por ejemplo *"Calentamiento Global"*
2. A partir del tema se seleccionó un documento que hablara del tema.
3. Una vez obtenido el documento base del conjunto, se analizó el contenido y se obtuvieron 2 documentos que hablaran de temas afines.
4. Con el conjunto de tres documentos establecido, se creó el conjunto de frases que serán usadas para testear. Las frases fueron diseñadas para que sean identificables en los documentos del conjunto, de forma que permitan probar diversos aspectos del problema (por ejemplo las distintas relaciones).
5. Se repiten estos pasos hasta tener cinco conjuntos de documentos.

Los tres documentos no relacionados, correspondientes al sexto subconjunto, fueron seleccionados del corpus de documentos Corin[18]. Este subconjunto de documentos será probado para todas las frases de los subconjuntos anteriores.

En el Cuadro 6.1 se muestran las frases seleccionadas.

Id	Grupo	Id	Frases de consulta
1	1	G1-1	*"problemas del calentamiento global"*
2	1	G1-2	*"el cambio climático"*
3	1	G1-3	*"efecto invernadero"*
4	1	G1-4	*"petróleo"*
5	2	G2-1	*"aumento en los precios"*
6	2	G2-2	*"política económica"*
7	2	G2-3	*"crisis financiera"*
8	2	G2-4	*"oferta y demanda"*
9	3	G3-1	*"gripe"*
10	3	G3-2	*"enfermedades contagiosas"*
11	3	G3-3	*"epidemias y pandemias"*
12	3	G3-4	*"muertes ocasionadas por enfermedades"*
13	4	G4-1	*"ciencia ficción"*
14	4	G4-2	*"el hombre del próximo milenio"*
15	4	G4-3	*"las tecnologías y ciencias del futuro"*
16	4	G4-4	*"el tercer milenio"*
17	5	G5-1	*"derechos humanos"*
18	5	G5-2	*"la democracia, la igualdad y la justicia"*
19	5	G5-3	*"obligaciones de los estados"*
20	5	G5-4	*"libertad"*

Cuadro 6.1: Frases de prueba usadas en las pruebas finales del sistema

En el Cuadro 6.2 se presenta informacíon sobre la composición del corpus utilizado para las pruebas.

Total de documentos	18
Oraciones	1339
Oraciones con offset	1279
Palabras	33770
Palabras con offset	14433
Tokens	37735
Offsets	74451

Cuadro 6.2: Datos estadísticos del corpus

6.1.3. Definición de los parámetros del sistema

El sistema desarrollado tiene múltiples parámetros que afectan su comportamiento. En esta sección presentaremos los parámetros más importantes, explicaremos su comportamiento y el valor elegido.

- ***sistema.usar_ventana_de_desambiguacion = false***
  Indica si se utilizará la desambiguación manual.

- ***freeling.tokens.UPDATE_TOKENS_OFFSETS = true***
  Indica si a los resultados de Freeling que no obtuvieron información de *offsets* le agregamos información obtenida por WordNet, en otras palabras, mejorar la identificación de Freeling expandiéndola con WordNet.

- ***algoritmos.analisis.USAR_TODOS_LOS_OFFSETS_DE_CONSULTA = false***
  Indica si se usarán todos los *offsets* de las palabras de la frase de consulta.

- ***algoritmos.analisis.MAX_DEEP_RELATION = 4***
  Indica hasta cuantas relaciones recursivas se analizarán.

- ***algoritmos.directoponderado.DIRECTO_PONDERADO_MAX_RESULT = 10***
  Indica la cantidad máxima de resultados (K) que retorna el *"Algoritmo de selección 1"*.

- ***algoritmos.directoponderado.tolerancia.minima=30***
  Indica la tolerancia mínima para el *"Algoritmo de selección 2"*.

- **Valores de ponderación para los diferentes tipos de relaciones:**
  - *algoritmos.ponderacion.LEMAS = 15*
  - *algoritmos.ponderacion.SINONIMOS = 15*
  - *algoritmos.ponderacion.HAS_HYPONYM_INVERSE = 8*
  - *algoritmos.ponderacion.HAS_HYPONYM = 10*
  - *algoritmos.ponderacion.NEAR_SYNONYM = 10*
  - *algoritmos.ponderacion.HAS_DERIVED = 4*
  - *algoritmos.ponderacion.HAS_HOLO_PART = 4*
  - *algoritmos.ponderacion.HAS_HOLO_MADEOF = 2*
  - *algoritmos.ponderacion.HAS_HOLO_MEMBER = 6*
  - *algoritmos.ponderacion.NEAR_ANTONYM = 5*
  - *algoritmos.ponderacion.BE_IN_STATE = 1*
  - *algoritmos.ponderacion.ROLE_AGENT = 1*
  - *algoritmos.ponderacion.HAS_XPOS_HYPONYM = 2*
  - *algoritmos.ponderacion.HAS_SUBEVENT = 1*
  - *algoritmos.ponderacion.XPOS_NEAR_SYNONYM = 2*
  - *algoritmos.ponderacion.CAUSES = 1*

- *algoritmos.ponderacion.HAS_MERO_MEMBER = 1*
- *algoritmos.ponderacion.HAS_MERO_PART = 1*
- *algoritmos.ponderacion.PERTAINS_TO = 1*
- *algoritmos.ponderacion.ROLE = 1*
- *algoritmos.ponderacion.ROLE_INSTRUMENT = 1*
- *algoritmos.ponderacion.ROLE_LOCATION = 1*
- *algoritmos.ponderacion.ROLE_PATIENT = 1*
- *algoritmos.ponderacion.HAS_DERIVED_INVERSE = 2*
- *algoritmos.ponderacion.HAS_HOLO_PART_INVERSE = 2*
- *algoritmos.ponderacion.HAS_HOLO_MADEOF_INVERSE = 1*
- *algoritmos.ponderacion.HAS_HOLO_MEMBER_INVERSE = 3*
- *algoritmos.ponderacion.BE_IN_STATE_INVERSE = 1*
- *algoritmos.ponderacion.ROLE_AGENT_INVERSE = 1*
- *algoritmos.ponderacion.HAS_XPOS_HYPONYM_INVERSE = 2*
- *algoritmos.ponderacion.HAS_SUBEVENT_INVERSE = 1*
- *algoritmos.ponderacion.XPOS_NEAR_SYNONYM_INVERSE = 2*
- *algoritmos.ponderacion.CAUSES_INVERSE = 1*
- *algoritmos.ponderacion.HAS_MERO_MEMBER_INVERSE = 1*
- *algoritmos.ponderacion.HAS_MERO_PART_INVERSE = 1*
- *algoritmos.ponderacion.ROLE_INSTRUMENT_INVERSE = 1*
- *algoritmos.ponderacion.ROLE_LOCATION_INVERSE = 1*
- *algoritmos.ponderacion.ROLE_PATIENT_INVERSE = 1*

- **Coeficientes de reducción de ponderación por nivel de profundidad:**
  - algoritmos.ponderacion.profundidad.1=1
  - algoritmos.ponderacion.profundidad.2=0.5
  - algoritmos.ponderacion.profundidad.3=0.25
  - algoritmos.ponderacion.profundidad.4=0.10
  - algoritmos.ponderacion.profundidad.5=0.0
  - algoritmos.ponderacion.profundidad.6=0.0
  - algoritmos.ponderacion.profundidad.7=0.0

6.1.4. Análisis manual de los documentos

Para poder comparar los resultados de la identificación de nuestro sistema, debemos antes analizar y marcar manualmente todos los documentos del corpus para cada frase de consulta. Esta es una tarea ardua y no exenta de errores. El análisis completo de cada documento se puede encontrar en el CD adjunto a este documento, ver Anexo G *Contenido del CD de entrega*.

El marcado de los documentos se realizó siguiendo las siguientes pautas:

1. Una oración en el documento está relacionada con una frase de consulta, si puede ser asociadas semánticamente sin considerar conocimientos de la realidad o del mundo.

2. Dada una frase de consulta, se marcarán en el documento todas aquellas oraciones que estén relacionadas con la frase.

3. Las oraciones relacionadas en el documento se marcarán con un tag de la forma **<idConsulta1, idConsulta2,...>** al final de la oración.

Los documentos marcados manualmente serán comparados con los resultados del sistema.

6.1.5. Análisis automático del corpus con Desambiguación Automática

Utilizando el sistema desarrollado se deberá procesar cada documento para las frases de consulta correspondientes a su grupo, utilizando para la desambiguación automática, el significado más frecuente.

6.1.5.1. Resultados Obtenidos

En la tabla de la Figura 6.1 se presentan los resultados generales de las pruebas usando desambiguación automática para los diferentes algoritmos de selección de oraciones explicados en la Sección 5.4.1.

Las gráficas de la Figura 6.1 comparan los valores de cantidad de oraciones identificadas correctamente (IC), la cantidad de oraciones que fueron identificadas incorrectamente (II) y la cantidad de oraciones que no fueron identificadas (NI) por los algoritmos.Una observación que se desprende es que el *Algoritmo de selección 2* es menos proclive a identificar oraciones incorrectas que los demás.

En la Figura 6.2 podemos ver cómo es la relación entre las oraciones identificadas correctas (IC) y las no identificadas (NI) para los distintos algoritmos. De estas gráficas se desprende que el *Algoritmo de selección 3* tiene la mejor relación, faltándole solo 22,55 % (136) de oraciones por identificar, lo sigue el *Algoritmo de selección 2* con un un 24,87 % (150) de oraciones no identificadas y por último el *Algoritmo de selección 1* con un 47,59 % (287).

	Algoritmo de selección 1			Algoritmo de selección 2			Algoritmo de selección 3		
Grupo	IC	II	NI	IC	II	NI	IC	II	NI
G1	73	27	49	90	42	32	92	82	30
G2	75	33	44	104	35	15	108	72	11
G3	72	42	28	75	11	25	73	28	27
G4	20	59	2	13	12	9	18	41	4
G5	70	37	158	168	70	60	166	115	62
GDNR	6	150	6	3	8	9	10	36	2
TOTALES	316	348	287	453	178	150	467	374	136

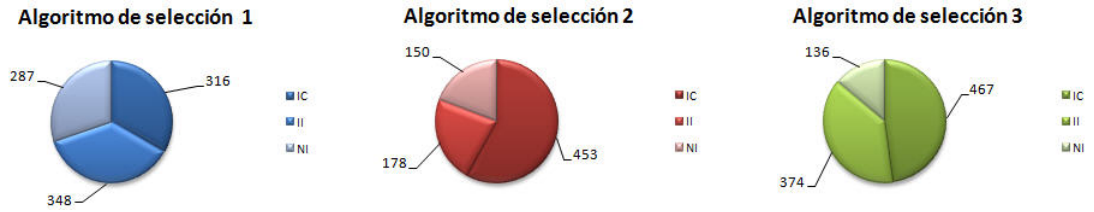


Figura 6.1: Resultados Generales usando Desambiguación Automática de significado

Figura 6.2: Comparación entre IC/NI

En la gráfica de la Figura 6.3 se puede comparar el funcionamiento de los distintos algoritmos para cada una de las variables clave (IC=Identificadas Correctas, II=Identificadas Incorrectas, NI=No identificadas). Se puede apreciar que el *Algoritmo de selección 3* es el que tiene un mayor número de oraciones identificadas correctamente y que el *Algoritmo de selección 2* es el que comete menos errores en la identificación.

6.1.5.2. Comportamiento según el largo de la frase de consulta

Un aspecto interesante de estudiar del comportamiento de los distintos algoritmos de selección de resultados, corresponde a cómo es su comportamiento de acuerdo a la cantidad de palabras que tiene la frase de consulta.

En las gráficas de la Figura 6.4 se puede apreciar cómo la *precision* del *Algoritmo de selección 1* y *Algoritmo de selección 3* decae al aumentar el número de palabras de la frase de consulta. A su vez se aprecia que aunque la *precision* del *Algoritmo de selección 2* se mantiene acotada el *recall* decae. Otro punto a destacar del *Algoritmo de selección 3*, si bien la cantidad de oraciones identificadas incorrectamente crece a la par de la cantidad de palabras de la frase, la cantidad de oraciones no identificadas disminuye de forma inversa.

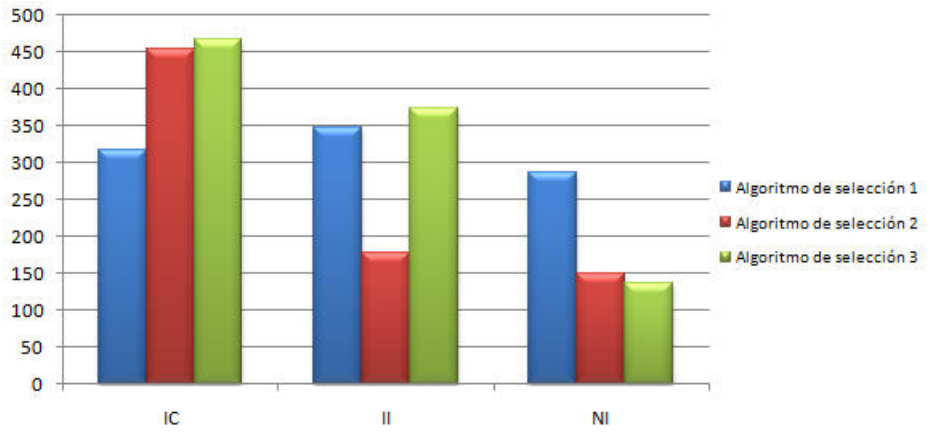


Figura 6.3: Comparación entre los distintos resultados

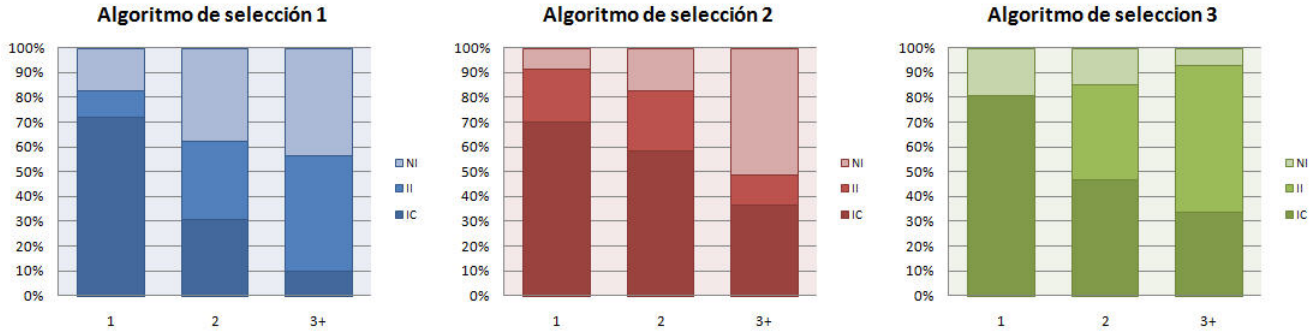


Figura 6.4: Comportamiento de los distintos algoritmos según la cantidad de palabras de las frases

6.1.5.3. Recall, Precision y F-measure

En el escenario de la recuperación de información, la *precision* está definida como el número de documentos relevantes recuperados dividido por el total de documentos recuperados. El *recall* se define como el número de documentos relevantes recuperados dividido el total de documentos que debieron ser recuperados.

Para que estas medidas se apliquen a nuestro sistema tenemos que verlas desde otro punto de vista, es por esto que definimos la *precision* y el *recall* de la siguiente forma:

Precision: Cantidad de oraciones correctamente identificadas sobre el total de oraciones identificadas.

$$Precision = \frac{oraciones\,correctas}{oraciones\,correctas\ +\ oraciones\,incorrectas}$$

Recall: Cantidad de oraciones correctas identificadas sobre el total de oraciones que tendrían que ser identificadas.

$$Recall = \frac{oraciones\,correctas}{oraciones\,identificables}$$

F-measure: Dado que el *precision* y el *recall* son antagónicos entre sí, se genera la necesidad de utilizar *F-measure* como fórmula de balance entre ambas.

$$F - measure = 2\frac{precision.recall}{precision + recall}$$

Los resultados para los distintos tipos de algoritmos de selección probados se muestran en el Cuadro 6.3 y se grafican en la Figura 6.5.

Algoritmo	**O.Correctas**	**O.Incorrectas**	**O.Identificables**	***Precision***	***Recall***	***F-measure***
Algoritmo de selección 1	316	348	603	47,59 %	52,40 %	49,88 %
Algoritmo de selección 2	453	178	603	71,79 %	75,12 %	73,42 %
Algoritmo de selección 3	467	374	603	55,53 %	77,45 %	64,68 %

Cuadro 6.3: Resultados de *Precision, Recall y F-measure*

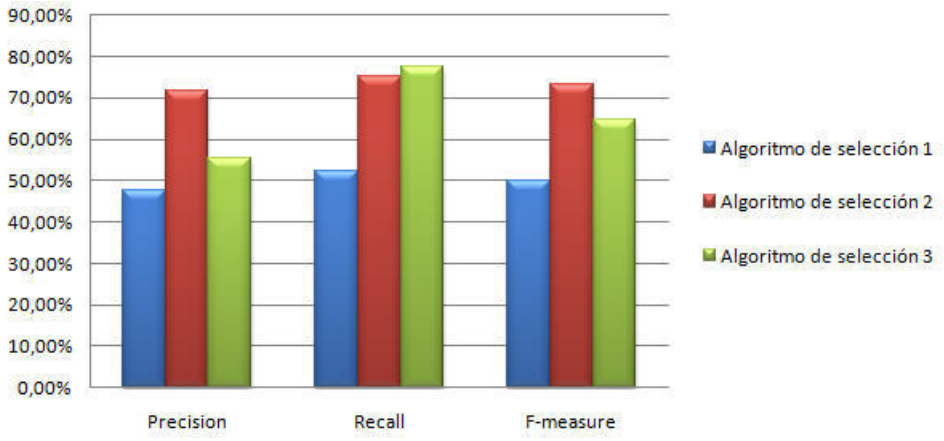


Figura 6.5: *Precision, Recall* y *F-measure* de los distintos algoritmos

6.1.6. Análisis automático del corpus con Desambiguación Manual

En esta sección se presentará los resultados obtenidos habiendo desambiguado manualmente la frase de consulta. Si bien los resultados son similares a los obtenidos en la desambiguación automática existen diferencias. La idea es luego poder compararlos y analizar los puntos fuertes y débiles de los dos métodos de desambiguación utilizados.

6.1.6.1. Elección de los significados de la expresión de consulta

Para realizar dicha tarea el sistema presenta para cada palabra de la frase de consulta sus posibles significados, de esta forma posibilita su elección por parte del usuario. El sistema utilizará los significados seleccionados para el procesamiento del documento.

Los significados seleccionados para cada una de las palabras de la frase de consulta pueden encontrarse en el Anexo C, *Pruebas del sistema*.

6.1.6.2. Resultados Obtenidos

	Algoritmo de Selección 1			Algoritmo de Selección 2			Algoritmo de Selección 3		
Grupo	IC	II	NI	IC	II	NI	IC	II	NI
G1	73	32	49	91	40	31	103	100	19
G2	71	40	48	107	37	12	106	75	13
G3	71	43	29	82	15	18	81	49	19
G4	21	62	1	15	10	7	18	35	4
G5	72	41	156	160	47	68	162	65	66
GDNR	3	118	9	2	5	10	2	36	10
TOTALES	311	336	292	457	154	146	472	360	131

Figura 6.6: Resultados Generales usando Desambiguación Manual de significado

En la tabla de la Figura 6.6 se presentan los resultados generales de las pruebas realizadas al sistema (usando desambiguación manual) .

En las gráficas de la Figura 6.6 se presenta una comparación de los valores de cantidad de oraciones identificadas correctamente (IC), identificadas incorrectamente (II) y no identificadas (NI) por los algoritmos.

Figura 6.7: Comparación entre IC/NI

En la Figura 6.7 podemos ver cómo es la relación entre las oraciones identificadas correctas (IC) y las no identificadas (NI).

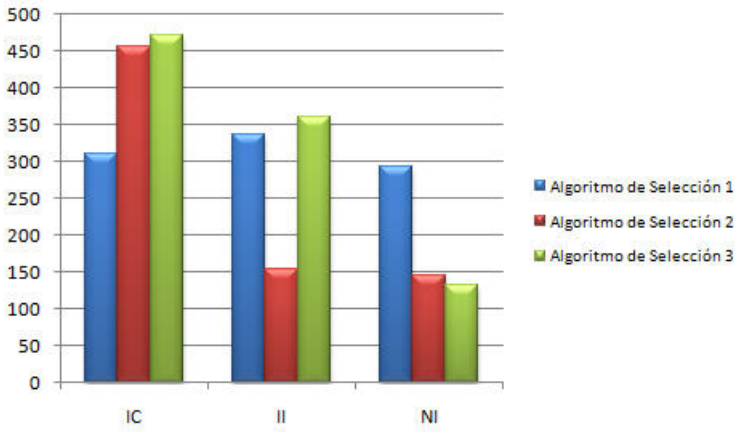


Figura 6.8: Comparación entre los distintos resultados

En la gráfica de la Figura 6.8 se puede comparar el funcionamiento de los distintos algoritmos para cada una de las variables claves (IC=Identificados Correctos, II=Identificados Incorrectos, NI=No identificados).

6.1.6.3. Comportamiento según el largo de la frase de consulta

Con respecto al comportamiento del sistema según el largo de la frase, utilizando desambiguación manual, obtuvimos los siguientes resultados.

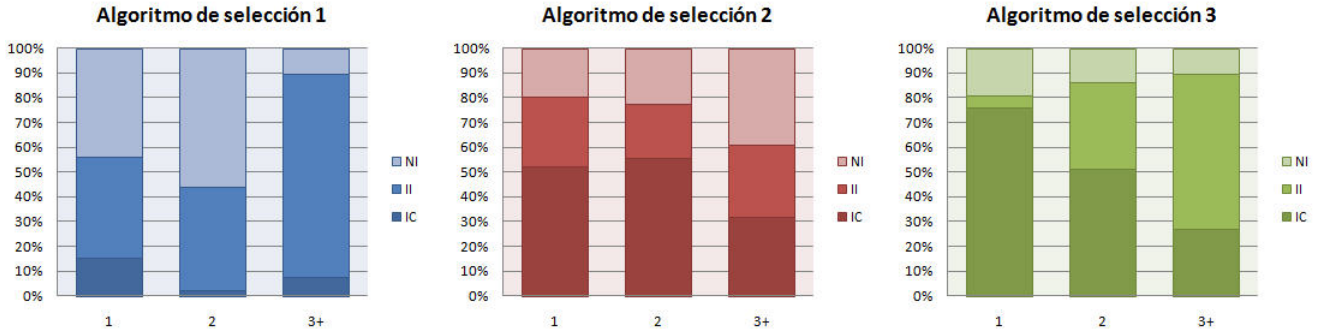


Figura 6.9: Comportamiento de los distintos algoritmos según la cantidad de palabras de las frases

En las gráficas de la Figura 6.9 se puede apreciar cómo la *precision* del *Algoritmo de selección 1* y *Algoritmo de selección 3* decae cuando más palabras tiene la frase. A su vez se aprecia que aunque la *precision* del *Algoritmo de selección 2* se mantiene acotada el *recall* disminuye.

Para el *Algoritmo de selección 3* se puede observar cómo aumenta el *recall* a la par de la cantidad de palabras de la frase.

Para el *Algoritmo de selección 2* se aprecia que si bien la cantidad de oraciones no

identificadas crece, la cantidad de oraciones incorrectas se mantiene en los mismos porcentajes.

6.1.6.4. Recall, Precision y F-measure

Los resultados para la *precision* y el *recall* se presentan en el Cuadro 6.4 y en la Figura 6.10.

Algoritmo	O.Correctas	O.Incorrectas	O.Identificables	*Precision*	*Recall*	*F-measure*
Algoritmo de selección 1	311	336	603	48,07 %	51,58 %	49,76 %
Algoritmo de selección 2	457	154	603	74,80 %	75,79 %	75,29 %
Algoritmo de selección 3	472	360	603	56,73 %	78,28 %	65,78

Cuadro 6.4: Resultados de *precision, recall* y *F-measure*

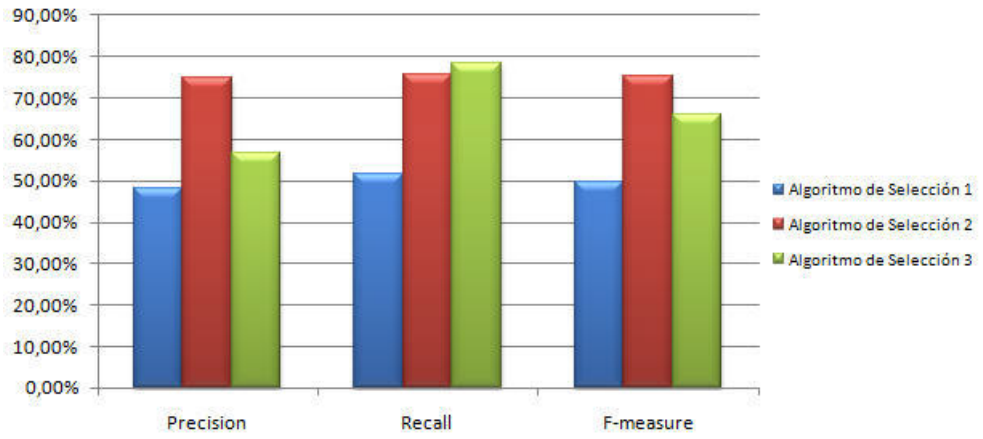


Figura 6.10: *Precision, recall* y *F-measure* de los distintos algoritmos

6.2. Comparación entre la desambiguación Manual y Automática

La tabla presentada en la Figura 6.11 presenta en primera instancia los resultados generales para la desambiguación manual (DM) y la desambiguación automática (DA), seguido de una línea en la cual se muestran las diferencias existentes para las diferentes variables estudiadas (IC, II, NI). También presenta el resultado del cálculo de *precision*, *recall y F-measure* para cada uno de los algoritmos de selección de resultados y para los distintos tipos de desambiguación. A éstos les sigue el cálculo de la mejora porcentual que se obtiene de la *precision*, *recall y F-measure* al utilizar desambiguación manual.

Algunas observaciones que pueden ser destacadas son:

	Algoritmo de selección 1			Algoritmo de selección 2			Algoritmo de selección 3			
TOTALES	IC	II	NI	IC	II	NI	IC	II	NI	
DM	311	336	292	457	154	146	472	360	131	
DA	316	348	287	453	178	150	467	374	136	
	-5	-12	5	4	-24	-4	5	-14	-5	DM - DA
DM	Precision	48,07%		Precision	74,80%		Precision	56,73%		
	Recall	51,58%		Recall	75,79%		Recall	78,28%		
	F-measure	49,76%		F-measure	75,29%		F-measure	65,78%		
DA	Precision	47,59%		Precision	71,79%		Precision	55,53%		
	Recall	52,40%		Recall	75,12%		Recall	77,45%		
	F-measure	49,88%		F-measure	73,42%		F-measure	64,68%		
	Precision	0,48%		Precision	3,00%		Precision	1,20%		
	Recall	-0,83%		Recall	0,66%		Recall	0,83%	DM-DA	
	F-measure	-0,12%		F-measure	1,87%		F-measure	1,10%		

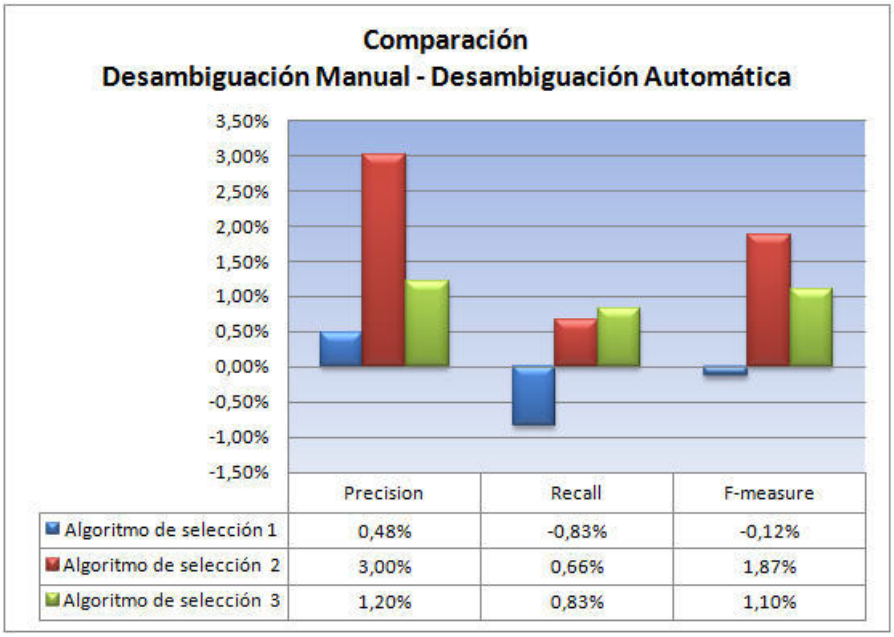
Figura 6.11: Tabla de comparación entre los resultados con desambiguación manual y automática de significado

- Salvo para el *Algoritmo de selección 1*, la cantidad de oraciones identificadas correctamente aumenta.
- Todos los algoritmos reconocen menos palabras incorrectas, especialmente el *Algoritmo de selección 2*.
- A los algoritmos 2 y 3, les quedan menos oraciones por identificar. Se da que es la relación inversa a la cantidad de identificaciones correctas que se lograron.

El hecho de que el *Algoritmo de selección 1* haya tenido una identificación de oraciones correctas inferior y un mayor número de oraciones no identificadas, puede explicarse de la siguiente forma: repasando como funciona el algoritmo de selección de *Algoritmo de selección 1* (explicado en la sección 5.4.1), vemos que este retorna las K mejores oraciones. Estas son las oraciones que relacionaron un mayor número de palabras con la expresión de consulta. Mediante el uso de la desambiguación manual, por lo general, se seleccionan palabras que tienen un mayor número de sinónimos e hiperónimos, lo que ocasiona que exista un número mayor de palabras candidatas a ser relacionadas. De esta forma el algoritmo de *Algoritmo de selección 1* tiene más chances de seleccionar las oraciones equivocadas, puesto que tiene un mayor número de oraciones para seleccionar.

En la gráfica de la Figura 6.12, se presenta la diferencia entre la *precision*, *recall* y *F-measure* para los diferentes algoritmos dependiendo del tipo de desambiguación. Algunos aspectos interesantes de destacar son:

- La *precision* de todos los algoritmos de selección de resultados aumentó, siendo el *Algoritmo de selección 2* quien tuvo una mejora más pronunciada, llegando a 3 %.
- Los algoritmos 2 y 3 mejoran también el *recall*, siendo el *Algoritmo de selección 3* quien tuvo la mejora más importante llegando casi al 1 %. El algoritmo 1 tuvo una caída en el *recall* causada por el aumento de oraciones no identificadas.



	Precision	Recall	F-measure
Algoritmo de selección 1	0,48%	-0,83%	-0,12%
Algoritmo de selección 2	3,00%	0,66%	1,87%
Algoritmo de selección 3	1,20%	0,83%	1,10%

Figura 6.12: Diferencia entre la *precision, recall* y *F-measure* según los diferentes métodos de desambiguación

Algunas conclusiones que podemos tomar luego de las comparaciones entre los distintos métodos de desambiguación son las siguientes:

- El método de desambiguación manual logra mejores resultados que la desambiguación automática especialmente cuando utilizamos significados muy especializados.
- La mejora de la desambiguación manual no es significativa. Una mejora del 3 % en la *precision* y 1 % en el *recall* no justifica la dificultad de la selección manual de los significados. Esta pobre mejora en los resultados puede deberse a que la base de datos Spanish WordNet es muy pequeña para lograr una mejor desambiguación por lo cual no se pueden brindar algunos significados. A esto se suma el hecho de que los conceptos están fuertemente unidos, por lo que aún cuando las palabras desambiguadas tengan significados distintos, alguna relación entre éstos puede ser encontrada por lo cual el sistema también la encuentra; por consiguiente, logra identificar casi lo mismo.

6.3. Detalle de errores

A continuación presentaremos algunos de los errores encontrados durante las pruebas que en su mayoría son generados por las herramientas utilizadas, explicaremos porqué ocurrieron y plantearemos posibles soluciones a los mismos.

- **Palabras compuestas**

Freeling reconoce un conjunto amplio de palabras compuestas. Son un grupo de palabras que en conjunto adquieren un significado que no puede ser determinado por el significado individual de cada una de las palabras que la componen. Por ejemplo, Freeling reconoce *"cambio climático"* como una única palabra compuesta por lo cual le asigna un único *offset*, si bien esto tiene sentido al tener la composición un significado más especifico que el brindado por las palabras por separado, también ocasiona que nuestro algoritmo no pueda relacionar las palabras individualmente. Una mejora al sistema en este punto sería que el sistema procese las palabras compuestas tanto como conjunto como en forma individual.

- **Ocurrencia de mayúsculas**

El uso "incorrecto" de mayúsculas puede ocasionar que Freeling reconozca un conjunto de palabras como una única palabra compuesta, que confunda verbos y adjetivos con nombres propios, entre otros inconvenientes. Esto dificulta la identificación de relaciones. Las frases *"Gases de Efecto Invernadero", "Defensores de los Derechos Humanos"* o *"ALERTA MUNDIAL POR GRIPE SUINA"* son algunos ejemplos.

- **Problemas de contexto**

Al considerar las oraciones como independientes de contexto, no se identifican oraciones que hablan del tema de la consulta, ya sea por omitir el sujeto o por correferencias a oraciones anteriores. Esto podría solucionarse ampliando el segmento de identificación a un párrafo completo (no solo una oración) o mediante un subsistema de resolución de correferencias.

- **Identificación de siglas**

En algunos documentos aparecen menciones a la frase de consulta pero a través de sus siglas, por ejemplo "Gases de efecto invernadero" aparece como "GEI" lo que dificulta la identificación. Esto podría ser resuelto mediante algún preprocesamiento del documento que genere una tabla de símbolos con las posibles abreviaciones o siglas que aparezcan en el documento.

- **Problemas con las relaciones léxicas**

Algunas relaciones léxicas ocasionan problemas en la identificación y como consecuencia que se retornen como correctas oraciones que no lo son. Un ejemplo complejo de esto puede ser *"combustibles alternativos"* que el sistema relacionó con la palabra *"petróleo"* y es sabido que los combustibles alternativos excluyen al petróleo.

- **Conocimiento del mundo**

Una persona es capaz de inferir información de la realidad y asociarla a una oración. Por ejemplo, para la oración *"El derretimiento de los glaciares puede ocasionar una elevación del nivel del mar",* un humano puede inferir que se está hablando de *"problemas ocasionados por el cambio climático"* sin que en el documento se haya nombrado explícitamente. Este problema es muy difícil de resolver y sería necesario utilizar métodos de aprendizaje automático y/o una base de datos de conocimiento general para lograr un acercamiento a la solución.

- **Palabras no existentes en WordNet**

Otro inconveniente que se encontró fue la poca cantidad de conceptos existentes y relacionados en Spanish WordNet, esto ocasiona la no identificación de relaciones entre la frase de consulta y el documento. Algunos ejemplos de estos son *"gripal"* y *"neogripe"* que no fueron relacionadas correctamente con la palabra *"gripe"*. Otros ejemplos son las palabras *"pandémica"* y *"pandémico"* que no fueron relacionadas con la palabra *"pandemia"*. Estos problemas pueden salvarse (en parte, porque siempre van a existir palabras que no estén indexadas) utilizando una versión de Spanish WordNet más actualizada y potente u otra base de datos léxica más completa.

7 Conclusiones

La idea de alcanzar una "inteligencia artificial" está todavía muy lejana. Esto es así porque hasta los pilares básicos necesarios para cumplir con este objetivo no han sido logrados, uno de éstos consiste en el entendimiento automático del lenguaje humano. Pero sí se han realizado trabajos apuntando a resolver problemas puntuales. Una dificultad que no puede ser pasada por alto es el hecho de que la gran mayoría de estos trabajos están principalmente enfocados para el idioma inglés. El trabajo realizado por nosotros pretende ser un aporte en este punto, enfocándonos en el idioma español.

7.1. Sobre el sistema

Para que el sistema realice la identificación debe analizar tanto la expresión de consulta como el documento íntegro. A través del análisis se pretende relacionar las palabras que conforman el documento con las palabras en la expresión de consulta. Para mejorar la capacidad de relacionamiento, se utiliza una base de datos léxica que permite no sólo relacionar palabras iguales sino también palabras vinculadas mediante otros tipos de relaciones existentes en el idioma, por ejemplo sinónimos, hiperónimos, hipónimos, merónimos, entre otras.

Consideramos que el sistema desarrollado cumple de forma satisfactoria con los objetivos planteados. El sistema resuelve correctamente los problemas de reconocimiento de oraciones, identificación de entidades con nombre, reconocimiento de locuciones, análisis de categoría gramatical, reconocimiento de los significados de las palabras, identificación del lema de una palabra, reconocimiento de sinónimos, hiperónimos, merónimos y otros tipos de relaciones, expansión de la consulta, descarte de palabras que no aportan información semántica, desambiguación automática y por último se logró una correcta identificación de segmentos relacionados temáticamente (teniendo en cuenta los valores de *precision* y *recall* de este tipo de sistemas).

Algunas características a destacar del sistema son:

- ***Altamente parametrizable***. El sistema tiene múltiples parámetros que permiten definir su comportamiento.
- ***Configurable dinámicamente.*** El sistema brinda la posibilidad de configurar parte de su comportamiento de forma dinámica.
- ***Permite múltiples tipos de desambiguación***. Desambiguación manual (DM), desambiguación automática (DA) o usar todos los posibles significados.

- ***Altamente modulado***. Los componentes del sistema fueron desarrollados como módulos, permitiendo la sustitución de componentes específicos, así como su utilización de forma independiente o en conjunto.

- ***Diferentes algoritmos de selección de resultados***. El sistema ofrece tres algoritmos distintos para la selección de resultados, cada uno con ventajas y desventajas. Además se puede incorporar un nuevo algoritmo de forma simple y rápida, utilizando la mayoría de los componentes y librerías de ayuda desarrolladas (por ejemplo, librerías que encapsulan los servicios de Freeling y WordNet).

- ***Integración con Lavinia.*** La separación en módulos y la utilización de UIMA permitió integrar nuestro sistema a la plataforma Lavinia quedando de esta forma accesible desde internet.

Para evaluar el comportamiento del sistema se utilizaron las variables de *precision*, *recall* y *F-measure* (variables utilizadas frecuentemente en sistemas similares).

		Algoritmo de selección 1	Algoritmo de selección 2	Algoritmo de selección 3
DM	Precision	48,07%	74,80%	56,73%
	Recall	51,58%	75,79%	78,28%
	F-Measure	49,76%	75,29%	65,78%
DA	Precision	47,59%	71,79%	55,53%
	Recall	52,40%	75,12%	77,45%
	F-measure	49,88%	73,42%	64,68%

Figura 7.1: *Precision*, *Recall y F-measure* usando desambiguación manual(DM) y desambiguación automática(DA)

En la Figura 7.1 se presenta un resumen de los resultados obtenidos por el sistema para los distintos algoritmos de selección de resultados, tanto para los métodos de desambiguación manual como automático. Se puede observar que el sistema llega a una *precision* de hasta un 74,80 % y un *recall* de 78,28 %.

Los resultados obtenidos para estas variables mostraron un comportamiento mejor al esperado, considerando el rendimiento de sistemas similares para el inglés[22, 23, 25, 24], los cuales alcanzan valores entre un 40 % y un 75 %. Igualmente es importante destacar que no se encontró ningún trabajo análogo al nuestro que nos permitiera realizar una comparación más adecuada.

Entre los problemas que fueron dejados para ser resueltos en futuros proyectos, debido a escasez de tiempo y recursos, se destacan la identificación de la raíz léxica de una palabra, el análisis de metáforas y metonimias, y el análisis de correferencias y resolución de anáforas.

7.2. Sobre el proyecto

Algunos aspectos que queremos destacar del proceso del desarrollo del proyecto son:

- Se trabajó con múltiples herramientas enfocadas al área de PLN (Freeling, WordNet, UIMA y Lavinia). Se estudiaron, analizaron y finalmente se integraron para lograr en conjunto la solución al problema.
- Se logró un mejor entendimiento de una gran cantidad de problemas del área.
- Se analizaron trabajos relacionados y se estudió como podían ser aplicados en nuestro sistema.
- Consideramos que es un valioso aporte la obtención de una licencia de Spanish WordNet con fines académicos. Licencia que queda disponible para el Grupo de PLN del InCo, para ser usada en futuros cursos, proyectos e investigaciones.
- Se construyó una herramienta extensible capaz de ser utilizada en proyectos de mayor envergadura. La arquitectura en componentes independientes permite que cada uno de éstos pueda ser integrado en otros proyectos.
  Otras herramientas que dejamos disponibles (a parte de los componentes) y que pueden ser utilizados independientemente son:
  - Librería de acceso a Spanish WordNet, brinda servicios de consulta y representación de datos para la base de datos.
  - Librería de utilidades lingüísticas de Freeling, brinda servicios para el análisis lingüístico de un texto.
  - Aplicación con interfaz gráfica desarrollada en Java-Swing destinada a la desambiguación manual del significado de un texto.
- Creemos que tanto la investigación realizada como el producto obtenido es un importante aporte al área, en particular, para el español. Esperamos que el sistema siga evolucionando, incorporando los recursos necesarios para poder seguir trabajando en los problemas que no pudieron ser abordados durante el transcurso de este proyecto.

Por último, queremos afirmar que estamos muy contentos y conformes con los resultados del trabajo, esperando haber contribuido en el desarrollo de estas problemáticas.

Glosario

Este apéndice tiene como objetivo definir conceptos básicos mencionados a lo largo de toda la documentación.

Ambiente de desarrollo Un entorno de desarrollo integrado o, en inglés, Integrated Development Environment (IDE), es un programa compuesto por un conjunto de herramientas para un programador. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

Anáfora Es la presencia en la oración de elementos que hacen referencia a algo mencionado con anterioridad. La anáfora es un tipo de deixis[1] que desempeñan generalmente los pronombres o adverbios.

Análisis sintáctico Proceso de los sistemas de extracción de información en el que se identifica la manera en que están construidas las oraciones para tratar de obtener información relevante.

Análisis sintagmático El análisis sintagmático consiste en determinar el papel que las palabras desempeñan dentro de cada sintagma: su núcleo y la función de las palabras que lo rodean (los determinantes, modificadores y complementos).

Anillo de sinónimos Es un conjunto de palabras o frases que son consideradas como equivalentes para la recuperación de información.

Antónimo [14]Palabra que expresa una idea opuesta o contraria a la expresada por otra palabra: *"vicio"* y *"virtud"* son palabras antónimas; el antónimo de *"claro"* es *"oscuro"*.

Arquitectura de software Corresponde al diseño de más alto nivel de la estructura de un sistema. Consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos.

[1]Del griego, significa señalar. Es la parte de la pragmática que está relacionada con las palabras que sirven para indicarnos cosas. Palabras como tú, hoy, aquí, esto, son expresiones deícticas, que nos sirven para señalar personas, situaciones, lugares, etc.

Artículo definido El determinante **el** (y sus variantes de género y número) se conoce como artículo definido.

Artículo indefinido Los determinantes **un**, **algún**, y **cierto** (y sus variantes de género y número) se conocen como artículos indefinidos.

Categoría gramatical [12] Se denomina *categoría gramatical* en la gramática tradicional a la modificación que las palabras variables de la oración presentan en su forma para expresar diferentes funciones gramaticales.

Conjunción [14] Parte invariable de la oración que denota la relación que existe entre dos palabras, sintagmas u oraciones, juntándolas o enlazándolas siempre gramaticalmente, aunque a veces signifique contrariedad o separación de sentido entre ellas. Por ejemplo *"y"*, *"ó"*.

Contracción Se denomina contracción a la forma abreviada mediante la cual se pronuncia y escribe en una sola palabra el encuentro de determinadas preposiciones con los artículos. En el idioma español se dan dos casos: *a + el = al* y *de + el = del.*

Corpus Conjunto de datos, textos u otros materiales sobre determinada materia que pueden servir de base para una investigación o trabajo.

Demostrativos Los determinantes **este**, **ese**, **aquel**, y sus variantes plurales se conocen como determinantes demostrativos.

Desambiguación del significado Es el proceso por el cual se logra identificar el significado de una palabra en un contexto.

Diagrama de Gantt [13]El diagrama de Gantt es una popular herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

Dominio Hace referencia a una específica área de interés o a una especifica área de conocimiento.

Entidades con nombre Son elementos atómicos en texto sobre categorías predefinidas como nombres de personas, organizaciones, localizaciones, expresiones de horas, cantidades, valores monetarios, porcentajes, etc. La tarea de reconocimiento de entidades con nombre (NER) busca localizar y clasificar estos elementos.

EuroWordNet [13] Es una base de datos multilingüística que une las distintas versiones de WordNet para varios lenguajes de las lenguas Europeas (Alemán, Italiano, Español, Holandés, Francés, Checo y Estonio). Cada una de las distintas versiones de WordNet está estructurada de igual forma que la versión

Americana[2] agrupando los términos en *synsets* y con las relaciones semánticas básicas que ocurren entre ellos. EuroWordNet a diferencia de la versión de Princeton agrega relaciones entre idiomas permitiendo relacionar palabras similares entre los diferentes idiomas.

Familia léxica Una familia léxica es el conjunto de palabras que comparte un mismo lexema o raíz lexica.

Gramática [16] Ciencia que estudia los elementos de una lengua y sus combinaciones.

Hiperónimo [14] Palabra cuyo significado engloba el de otras más específicas: *"mueble"* es un hiperónimo de *"silla"*. Semánticamente, un hiperónimo no posee ningún rasgo semántico, o sema, que no comparta con su hipónimo, mientras que este posee rasgos semánticos que lo diferencian de aquel.

Hipónimo [14] Palabra cuyo significado es más específico que el de otra en la que está englobada: *"minuto"* es un hipónimo de *"tiempo"*. Posee todos los rasgos semánticos, o semas, de otra más general, su hiperónimo, pero que añade en su definición otros rasgos semánticos que la diferencian de la última.

Homonimia La homonimia es una relación léxica que se establece entre dos homónimos, es decir, palabras que presentan identidad formal (fónica o gráfica) pero diferencia en el significado.

Inflexión gramatical La flexión es la alteración que experimentan las palabras mediante morfemas constituyentes según el significado gramatical o categoría para expresar sus distintas funciones dentro de la oración y sus relaciones de dependencia o de concordancia con otras palabras o elementos oracionales.

Metáfora [14] Figura consistente en usar una palabra o frase por otra, estableciendo entre ellas un símil no expresado: *"Tus ojos son dos luceros"* significa que tienes los ojos brillantes o iluminados.

Merónimos Se denomina merónimo a la palabra cuyo significado constituye una parte del significado total de otra palabra, denominada ésta holónimo. Por ejemplo, "dedo" es merónimo de "mano" y "mano" es merónimo de "brazo"; a su vez, "brazo" es holónimo de "mano" y "mano" es holónimo de "dedo". Por lo tanto: X es merónimo de Y si X forma parte, es una sustancia ó es un miembro de Y.

Metonimia [14] Figura consistente en designar una cosa con el nombre de otra con la que guarda una relación de causa a efecto, autor a sus obras, etc. Por ejemplo *"Tiene quince primaveras"* hace referencia a que tiene quince años.

Lavinia [17] Es un ambiente basado en UIMA para Procesamiento de Lenguaje Natural, desarrollado por el GPLN del InCo - Fing - UdelaR.

[2]La versión Americana de WordNet es la primera versión desarrollada por Miller en 1990 en la Universidad de Princeton.

Lema Abstracción, a partir del haz de rasgos flexivos de una palabra, que representa a esta como forma canónica. Un lema representa un conjunto de palabras con la misma raíz, misma categoría léxica (tipo de palabra) y mismo significado.

Lenguaje [16] Se considera como un conjunto de oraciones, que usualmente es infinito y se forma con combinaciones de palabras del diccionario. Es necesario que esas combinaciones sean correctas (con respecto a la sintaxis) y tengan sentido (con respecto a la semántica).

Lingüística Ciencia cuyo objetivo es el estudio teórico del lenguaje y las leyes que lo gobiernan.

Lingüística computacional Hace referencia al campo multidisciplinario de la lingüística y de la computación. Es el estudio científico del lenguaje con el fin de elaborar modelos de éste o de ciertos fenómenos específicos. Involucra a lingüistas, informáticos, lógicos, psicólogos cognitivos.

Locución En gramática, una locución es el grupo estable de dos o más palabras que funciona como una unidad léxica con significado propio, no derivado de la suma de significados de sus componentes.

"de la cabeza a los pies"

"ácido sulfúrico"

"desde que el mundo es mundo"

Oración Palabra o conjunto de palabras con sentido gramatical completo.

Parsing (parseo) [21] Es el proceso que toma una entrada y genera alguna clase de salida estructurada.

Polisemia Se denomina polisemia a la capacidad que tiene una sola palabra para expresar distintos significados.

Posesivos Los determinantes **mi**, **tu**, **su**, y sus variantes plurales se conocen como (adjetivos) posesivos.

Pronombre Parte de la oración que puede ocupar el lugar de un nombre o hacer alusión a él. Sustituye en ocasiones, aunque no siempre, a un sustantivo, al que se denomina antecedente. Existen diferentes clases de pronombres, cada una representante del tipo de sustantivo al que sustituye. Las clases de pronombres para el idioma español son: personales, numerales, demostrativos, interrogativos, posesivos, exclamativos, indefinidos y relativos.

Proposición También conocido como cláusula es una unidad gramatical con estructura oracional pero que carece de independencia sintáctica, semántica, y fonológica. Esta falta de autonomía es la principal diferencia respecto de la oración, unidad completa e independiente. Una proposición necesita relacionarse con otras proposiciones para formar una oración.

Raíz léxica Se denomina raíz léxica o lexema al monema que posee un significado autónomo e independiente y constituye la parte invariable de una palabra.

Semántica Parte de la lingüística que estudia el significado de las expresiones lingüísticas.

Sinónimo [14] Vocablos o expresiones que tienen una misma o muy parecida significación: *"esposos"* y *"cónyuges"* son términos sinónimos.

Sintagma Un sintagma es una unidad de función sintáctica o un tipo de constituyente. Se trata de una agrupación lineal de palabras vinculadas entre sí en torno a un núcleo y organizadas según las reglas gramaticales de una lengua. El núcleo sintáctico es la palabra que aporta las características básicas del sintagma.
El sintagma es una unidad sintáctica inmediatamente superior al constituyente no-sintagmático. Todas las oraciones pueden descomponerse en sintagmas; incluso, la oración misma puede ser considerada como un macrosintagma.

Sintagma nominal También conocido como frase nominal, un sintagma nominal describe a un grupo de palabras que forma un constituyente maximal cuyo núcleo está formado por un sustantivo, adjetivo sustantivado o pronombre.
En la lengua española, un sintagma nominal puede actuar como sujeto agente o paciente, atributo, complemento directo, complemento predicativo, aposición, vocativo o complemento circunstancial. Si el núcleo es un pronombre, puede funcionar además como complemento indirecto y complemento directo de persona.

Synset Es un grupo de elementos de información que son considerados semánticamente equivalentes para propósitos de recuperación de información. Por ejemplo las palabras *"auto"*, *"coche"*, *"automóvil"*, *"vehículo"* forman parte de un mismo *synset*.

Taxonomía Ciencia que se ocupa de los principios, métodos y fines de la clasificación. Clasificación que se realiza según esta ciencia, en especial la que ordena, jerarquiza y nombra, dentro de la biología, los seres vivos.

Tesauro Un tesauro es una lista que contiene los "términos" empleados para representar los conceptos, temas o contenidos de los documentos, con miras a efectuar una normalización terminológica que permita mejorar el canal de acceso y comunicación entre los usuarios y las Unidades de Información (Entienda se unidad de información como: biblioteca, archivo o centros de Documentación).

Testing Las pruebas de software, testing o beta testing es un proceso usado para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador o videojuego. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas. *"El testing puede probar la presencia de errores pero no la ausencia de ellos"*, E. W. Dijkstra.

WordNet [13] WordNet es una enorme base de datos léxica del idioma inglés. Agrupa las palabras en conjuntos de sinónimos llamados *synsets*, proporcionando definiciones cortas y generales, y almacenando las relaciones semánticas entre estos conjuntos de sinónimos.

Bibliografía

[1] Analizador lingüístico Freeling
http://garraf.epsevg.upc.es/freeling
Último acceso: 14/11/2009

[2] WordNet
http://wordnet.princeton.edu
Último acceso: 14/11/2009

[3] EuroWordNet
http://www.illc.uva.nl/EuroWordNet
Último acceso: 14/11/2009

[4] Spanish WordNet
Último acceso: 14/11/2009

[5] Apache UIMA
http://incubator.apache.org/uima
Último acceso: 14/11/2009

[6] Apache Tomcat
http://tomcat.apache.org
Último acceso: 14/11/2009

[7] Lavinia - Grupo de Procesamiento de Lenguaje Natural - UdelaR
http://www.fing.edu.uy/inco/grupos/pln/lavinia.html
Último acceso: 14/11/2009

[8] Java Swing
http://java.sun.com/javase/6/docs/technotes/guides/swing
Último acceso: 14/11/2009

[9] TREC (*Text REtrival Conference*)
http://trec.nist.gov
Último acceso: 14/11/2009

[10] NIST (*National Institute of Standards and Technology)*
http://www.nist.gov
Último acceso: 14/11/2009

[11] MUC (*Message Understanding Conference*)
http://www-nlpir.nist.gov/related_projects/muc
Último acceso: 14/11/2009

[12] Enciclopédia Encarta
http://es.encarta.msn.com
Último acceso: 14/11/2009

[13] Wikipedia
http://www.wikipedia.org
Último acceso: 14/11/2009

[14] WordReference
http://www.wordreference.com
Último acceso: 14/11/2009

[15] Wiki Lengua del español
http://www.wikilengua.org
Último acceso: 14/11/2009

[16] Real Academia Española.
http://www.rae.es/
Último acceso: 14/11/2009

[17] Grupo de Procesamiento de Lenguaje Natural - UdelaR
http://www.fing.edu.uy/inco/grupos/pln
Último acceso: 14/11/2009

[18] Corin
http://www.fing.edu.uy/inco/grupos/pln/publicaciones/Uruguay-Grassi-Wonsever-et-al_2_.pdf
Último acceso: 14/11/2009

[19] Proyecto RicoTerm 2
http://ricoterm.iula.upf.edu/2/
Último acceso: 14/11/2009

[20] Design Patterns. Elements of Reusable Object Oriented Software - *Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (2007)*

[21] Speech and Language Processing - *Daniel Jurafsky, James H. Martin - Prentice-Hall (2000)*

[22] Automatic Topic Identification Using Ontology Hierarchy - *Sabrina Tiun, Rosni Abdullah, Tang Enya Kong (2001)*
http://utmk.cs.usm.my/pdf/CICLing01.pdf
Último acceso: 14/11/2009

[23] Knowledge-based Automatic Topic Identification - *Chin-Yew Lin, Departamento de ingeniería Eléctrica, Universidad del Sur de California, Los Angeles, Estados Unidos (1995)*
http://acl.ldc.upenn.edu/P/P95/P95-1046.pdf
Último acceso: 14/11/2009

[24] The Effect of Using Hierarchical Classifiers in Text Categorization - *Stephen D'Alessio, Keitha Murray, Robert Schiaffino (2000)*
http://www.iona.edu/academic/artsscience/departments/computerscience/faculty/FacultyPublications/riao2000New.doc
Último acceso: 14/11/2009

[25] Using Frequently Occurring Words to Identify the Subject of a Document - *Offer Drori, Universidad hebrea de Jerusalén (2001)*
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.3339&rep=rep1&type=pdf
Último acceso: 14/11/2009

[26] Using WordNet for Word Sense Disambiguation to Support Concept Map Construction - *Institute for Human and Machine Cognition - Universidad EAFIT (2003)*
http://cmap.ihmc.us/Publications/ResearchPapers/SPIRE-2003-UsingWordnetforWordSenseDisambiguationtoSupportConceptMapConstruction.pdf
Último acceso: 14/11/2009

[27] Information Extraction: Techniques and Challenges - *Ralph Grishman, Departamento de Ciencias de la Computación, Universidad de Nueva York, Estados Unidos (1997)*
http://web.njit.edu/~ql23/teaching/cis634FTF/notes/Information_extraction_paper.pdf
Último acceso: 14/11/2009

[28] Identifying Topics by Position - *Chin-Yen Lin, Eduard Hovy. Departamento de Ciencias de la Computación, Universidad del Sur de California, Los Angeles, Estados Unidos (1997)*
http://www.mariapinto.es/ciberabstracts/Articulos/74.pdf
Último acceso: 14/11/2009

[29] Algoritmo de resolución de la anáfora pronominal en diálogos - *Patricio Martínez Barco, Grupo de Investigación en Procesamiento del Lenguaje y Sistemas de Información, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante (1999)*
http://www.sepln.org/revistaSEPLN/revista/24/24-articulo8.pdf
Último acceso: 14/11/2009

[30] Resolución de anáfora pronominal para el idioma español usando el método de conocimiento limitado - *Grigori Sidorov, Omar Olivas Zazueta Laboratorio de Lenguaje Natural y Procesamiento de Texto, Centro de Investigación en Computación, Instituto Politécnico Nacional (2006)*
http://ccc.inaoep.mx/~tec_lenguaje06/articulos/TLH06-paper11.pdf
Último acceso: 14/11/2009

[31] Ontología por Tom Gruber - *Encyclopedia of Database Systems (2007)*
http://tomgruber.org/writing/ontology-definition-2007.htm

[32] Definición de ontología como especificación del conocimiento - *Chantal Pérez (2002)*
http://elies.rediris.es/elies18/531.html
Último acceso: 14/11/2009

[33] Evaluación de Buscadores Web - *Marta Torres Magán, Universidad Carlos III de Madrid*
http://usuarios.lycos.es/webbuscadores/evaluacion_buscadores.pdf
Último acceso: 14/11/2009

[34] Definicion de EI brindada por Javier Aragón Zabalegui - Universidad Carlos III de Madrid
http://extraccioninformacion.iespana.es
Último acceso: 14/11/2009

[35] Multilingual Ontology-Based Lexicon for News Filtering - The TREVI Project - *Weigand, H. (1997)*

[36] Integrating Medical Terminologies with ONIONS Methodology - *Steve, G, A. Gangemi, D. Pisanelli (1998)*

[37] Formal Ontologies and Information Systems - *Guarino, N. (1998)*

[38] Learning to Extract Text-Based Information from the World Wide Web - *Stephen Soderland (1997)*

Nomenclatura

ACE Automaric Content Extraction evaluation

DLL Dynamic Linking Library

DUC Document Understanding Conferences

EAGLES Expert Advisory Group on Language Engineering Standards

EI Extracción de Información

JAR Java ARchive

JNI Java Native Interface

MUC Message Understanding Conferences

NER Named Entity Recognition

NIST National Institute of Standards and Technology

PLN Procesamiento de Lenguaje Natural

RAE Real Academia Española

RI Recuperacion de Informacíon

TAC Text Analysis Conference

TREC Text REetrieval Conference

UIMA Unstructured Information Management Applications

XML Extensible Markup Language

UNIVERSIDAD DE LA REPÚBLICA

Facultad de Ingeniería

InCo - Instituto de Computación

ANEXOS

"Sistema de Identificación de Segmentos Relacionados Temáticamente"

Tesis de grado presentada por

Martín Barreto - Ricardo Bedat

Docentes

Msc. Ing. Juan José Prada

Ing. Aiala Rosá

Montevideo, Uruguay - 2009

Índice general

A. Estudio del estado del arte

Este apéndice tiene como objetivo complementar y extender el estudio del estado del arte presentado de forma general en el capítulo 3, *Marco teórico*. Se presenta diferentes investigaciones en el área de Extracción de Información, principalmente trabajos relacionados con la determinación de la temática de documentos. También se presentan las principales herramientas utilizadas con el fin de poder identificar relaciones semánticas entre frases o porciones de un documento.

Durante el desarrollo de este proyecto, fue necesario realizar un estudio de las temáticas abarcadas por el mismo. Partiendo de una idea motivadora y escaso conocimiento en el área, el devenir de semanas y meses trajo consigo el estudio sistemático y progresivo de las distintas áreas de conocimiento relacionadas.

A.1. Trabajos de investigación

A.1.1. Resumen sobre extracción de información

Tema del paper

El resumen realizado por Damiano Spina Valenti, presenta los últimos avances en el campo de EI y describe las principales tareas o etapas de un proceso de EI. También presenta las conferencias que se han dedicado y se dedican a evaluar estos sistemas.

Resumen

Como muchos otros autores, se presenta la EI como un proceso de rellenar los campos de una base de datos (escenario de extracción) a partir de texto no estructurado o en formato libre. Básicamente consiste en la obtención de información estructurada a partir de información en lenguaje natural.

A este proceso lo divide en 5 tareas: preprocesamiento, clasificación, asociación, normalización y deduplicación.

Preprocesamiento Se realizan una serie de operaciones que facilitan el procesamiento posterior. Por ejemplo separar el texto en párrafos, oraciones (*splitters*), unidades léxicas (*tokenizers*), análisis morfosintáctico, *stemming*, lematización, etc.

Clasificación Determina qué campo del escenario de extracción hay que rellenar para cada segmento de texto. Esta tarea también es conocida como reconocimiento de entidades nombradas (*Named Entity recognition*) y se puede realizar con modelos

estadísticos basados en autómatas de estados finitos, como los modelos ocultos de Markov (*Hidden Markov Models*). También se puede utilizar modelos de probabilidad condicional como los modelos de máxima entropía de Markov y los campos aleatorios condicionales (*condition random fields*).

Asociación La asociación agrupa los campos que pertenecen a un mismo registro. También se conoce como extracción de relaciones entre entidades. Para conseguirlo se realiza un análisis sintáctico del documento y una interpretación semántica. Este análisis se realiza de forma parcial dado que hacerlo de forma total tiene una serie de inconvenientes (coste, incompletitud, ambigüedad, etc). Básicamente consiste en generar sintagmas y resolver dependencias entre los mismos a partir de segmentos de textos clasificados. Estas dependencias se resuelven desde dos perspectivas distintas:

- Usando técnicas de encaje de patrones. Las cuales hacen uso de reglas de asociación que permiten identificar propiedades de entidades, también relaciones y eventos entre los mismos.

- Examinando las relaciones gramaticales. Para esto se definen un conjunto de relaciones entre entidades, como complemento circunstancial de tiempo o lugar.

Deduplicación Se elimina información redundante para no obtener registros duplicados en la base de datos. Para esta tarea se utilizan técnicas de análisis de discursos, que consisten en resolución de correferencias de anáforas.

Los sistemas de EI que presentan la información como plantillas parcialmente rellenas, utilizan métodos de mezclado.

Los sistemas que trabajan con formas lógicas usan procedimientos de interpretación semántica.

Normalización Se transforma la información extraída a un formato estándar, con el objetivo de tratar de forma sistemática segmentos equivalentes. Por ejemplo *"12 de noviembre del 2006"* y *"12/11/2006"*.

A.1.2. Information Extraction: Techniques and Challenges [27]

Tema del paper

Presenta la EI como el proceso de identificación de instancias (ocurrencias) para una clase particular de eventos o relaciones de un texto en lenguaje natural, así como también la extracción de atributos de interés pertenecientes al evento o relación.

Luego presenta la arquitectura para un sistema de EI (ver Figura A.1), presentando los componentes y la estructura del mismo.

Resumen

Se presenta un flujo para un sistema de EI, el mismo consta de 3 partes:

- Un análisis del texto local, en el cual el sistema extrae *"hechos"* individuales.
- Un proceso de integración de los *"hechos"* anteriores, utilizando inferencias, generando nuevos *"hechos"* o *"hechos"* más complejos.
- Por último, un proceso que transforme al formato de salida hechos seleccionados.

Los hechos individuales son extraídos mediante un conjunto patrones. Dado la naturaleza del lenguaje natural, estos patrones no pueden estar expresados en función de una secuencia de palabras, normalmente estos patrones son formulados en función de componentes y relaciones. Este primer proceso puede incluir análisis léxico (por ejemplo asignar la categoría gramatical mediante un análisis morfológico o buscar los sinónimos u otra relación semántica del concepto, etc.) y un proceso de reconocimientos de nombres. Este análisis de nombres podría abarcar el reconocimiento de nombres propios y otras estructuras léxicas como fechas, números, etc.

Luego también dentro del análisis local tenemos un procesamiento sintáctico. El último paso descripto en el análisis local del documento es aplicar los patrones que están basados en los metadatos inferidos en los pasos anteriores.

Podemos notar que solamente el último paso depende de qué tipo o información el sistema debe extraer, los pasos anteriores a éste son altamente genéricos y podrían utilizarse en casi cualquier sistema de extracción de información.

El proceso de integración trabaja con los hechos generados por la fase anterior. Básicamente analiza y relaciona los hechos inferidos de todo el documento. Esta etapa resuelve las relaciones de correferencias, que se dan por el uso de pronombres que hacen referencia al mismo nombre o evento.

Análisis Léxico En esta etapa divide el documento en oraciones y en palabras. Luego a cada palabra se le determina la categoría gramatical y otras características. Este proceso puede hacer uso de diccionarios de propósito general y diccionarios especiales, como diccionarios de empresas, de lugares, de nombres propios comunes, etc.

Reconocimiento de Nombres Busca localizar y clasificar elementos atómicos, como nombres propios, organizaciones, localizaciones, expresiones de horas, cantidades, valores monetarios, porcentajes, etc. Los nombres aparecen de forma frecuente en la mayoría de los textos e identificarlos es de gran ayuda para fases futuras del procesamiento.

Los nombres son identificados mediante un conjunto de expresiones regulares (patrones) basadas en la categoría gramatical, la estructura sintáctica, características ortográficas, etc.

Algunos nombres, por ejemplo el nombre de la empresa General Motors no tiene

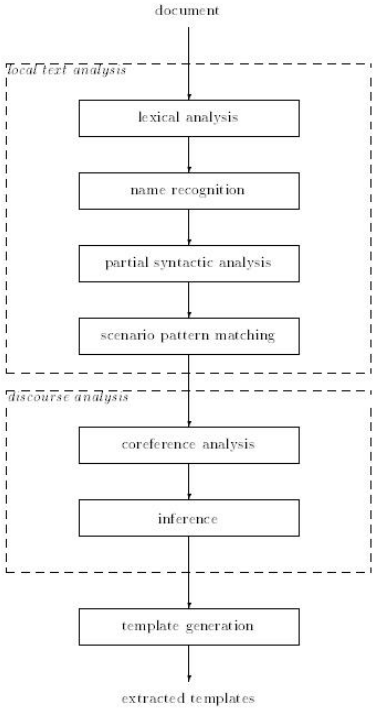


Figura A.1.: Arquitectura presentada por Ralph Grishman (ver A.1.2)

ningún Sufijo (Inc, Corporation, Associates, Bank), ningún título que lo preceda (por ejemplo Sr.), etc. En estos casos es útil tener un diccionario con las principales empresas a reconocer.
El reconocimiento de nombres incluye también la identificación de correferencias (alias de un nombre). La resolución de correferencias es de gran ayuda para poder clasificar los nombres, un ejemplo que presenta el paper es: si se encuentra "Hummble Hopp repoted ... " en este caso no se sabe si Hummble Hopp es una persona o una empresa. Pero si luego se encuentra Mr. Hopp esa ambigüedad se puede resolver.

Estructura Sintáctica Estudiar la estructura sintáctica simplifica la resolución en la extracción de los "hechos" en etapas posteriores. Esto se debe a que los argumentos que se quiere extraer corresponden a nombres en el texto *"noun phrases"* y las relaciones que se desea extraer a menudo corresponden a relaciones o funciones gramaticales *"grammatical functional relations"*.
Se menciona que una análisis completo de la estructura sintáctica es una tarea muy difícil. Por esta razón algunos sistemas no tienen una etapa separada de análisis sintáctico. Estos sistemas infieren estructuras solamente cuando están completamente seguro de ello, ya sea por una relación sintáctica o semántica[1].

Scenario Pattern Matching En esta etapa se extrae los eventos y las relaciones re-

[1]Para entender los detalles del sistema particular del autor ver [27].

levantes en el contexto, para esto se utiliza toda la información procesada por las etapas anteriores. Todo el procesamiento anterior fue realizado para facilitar y ayudar en esta etapa. Básicamente todos los macheos reconocidos por etapas anteriores se utilizaron como información para construir macheos más grandes. De esta forma se extrae la información de manera *button-up*, a partir de pequeñas identificaciones atómicas se identifica (relaciona) información con un nivel semántico mayor. El texto menciona un ejemplo en el cual identifica una empresa, y una persona, y luego en esta etapa tiene un patrón que relaciona la persona a la compañía. Acá se puede notar que las reglas de macheo a este nivel utilizan como elementos (compañía, persona) macheos realizados en etapas anteriores.

Análisis de correferencias Esta fase identifica frases que referencian al mismo objeto. Básicamente lo que resuelve son las anáforas[2] dadas por nombres y pronombres.

Inferencing and Event Merging En algunas situaciones, la información parcial de algunos eventos o relaciones está distribuida en muchas oraciones, en estos casos la información necesita ser integrada antes de generar los templates[3]. También existen casos en que la información está representada de una manera implícita y se necesita un proceso de inferencia que la convierta o explicite.
Las inferencias pueden ser implementadas mediante reglas[4].

Al desarrollar el sistema utilizaron un conjunto de corpus y de templates de salida esperados de entrenamiento, luego para evaluar los resultados se utilizaron un nuevo conjunto de documentos y sus templates resultantes.
Para evaluar el sistema utilizaron dos fórmulas, *precision* y *recall*.

$$Precision = \frac{N\,correctas}{N\,respuestas}$$

$$Recall = \frac{N\,correctas}{N\,clave}$$

Siendo *N correctas* el número de respuestas correctas, *N respuestas* el número de respuestas obtenidas por el sistema desarrollado, *N clave* el número de respuestas correctas esperadas (realmente correctas).

Otra fórmula utilizada que combina las dos anteriores es:

$$F = \frac{(2 * Precision * Recall)}{(Precision + Recall)}$$

[2]Ver anáfora en Glosario.
[3]Salida del sistema de EI estructurada que contiene campos.
[4]Predicados (lógica de predicados).

Conclusiones de los autores

Una de las diferencias más importantes de los sistemas de extracción de información es el nivel de procesamiento de análisis sintáctico que realizan. La importancia del análisis sintáctico está dada porque los eventos o relaciones que se quieren identificar en la etapa de *"Inferencing and Event Merging"* están casi siempre relacionados mediante relaciones gramaticales, esto genera que esta última etapa mencionada sea más simple y exacta.

También hace referencia a que un análisis sintáctico completo del texto es muy costoso en tiempo y difícil de realizar. Esto se debe a que tareas como la asignación de modificadores o determinar el alcance de las conjunciones son muy dificultosas, además se tendría que utilizar información global (restricciones) para resolver ambigüedades locales. Dado que al realizar un sistema de extracción de información uno está interesado en algún suceso en particular, en las relaciones gramaticales relevantes en el contexto, por lo cual realizar un análisis estructural de todo el documento no tendría ningún valor agregado e incrementaría los tiempos de procesamiento. Como conclusión para un sistema de extracción de información, realizar un análisis sintáctico parcial del corpus ya es suficiente, la estructura deberá crearse con una alta tasa de certeza y usando información local.

Es importante mencionar que el análisis sintáctico no sólo genera la estructura sintáctica sino que también tiene un rol de *regularizador* de la misma. Por ejemplo si tenemos diferentes formas de expresar lo mismo desearíamos tener una regla para identificar todas las formas que son o significan lo mismo, estas formas de expresar lo mismo son mapeadas para la misma estructura simplificando luego la etapa de *"scenario pattern matchine"*.

```
IBM contrato Harry
Harry fue contratado por IBM
Harry, quien fue contratado por IBM.
```

Si en la etapa que analiza la estructura sintáctica no se realiza el rol de regularizador se tendrá que tener diferentes *"scenario pattern"* para cada forma sintáctica.

Los sistemas SEI FASTUS y NYP Prteus presentaron una solución al respecto definiendo *metarules* o *rule schemata.* Básicamente son métodos para escribir un patrón y obtener los patrones necesarios para poder identificar ese patrón en varias formas sintácticas de una oración.
Un ejemplo de esto sería: Si el usuario ingresa la información mostrada abajo

```
Subject(sujeto) = empresa
verbo = contratar
object = persona
```

El sistema retorna:

```
Empresa contrató a persona.
```

Persona es contratada por *empresa*.
Empresa, la cual contrató a *persona*.
Persona, quien fue contratada por *empresa*.

Los patrones también tiene que tener en cuenta los modificadores que pueden aparecer entre los elementos de la oración.

Empresa **ayer** contrató a *persona*.

Como conclusión de esta parte, el uso extenso del análisis parcial es una consecuencia del estado de los analizadores completos. Igualmente los últimos van evolucionando de manera acelerada y mejorando en cuestiones de performance. Si estos sucesos persisten será necesario volver a evaluar si utilizar o no analizadores completos (*full sentence parsing*).

A.1.3. Automatic Topic Identification Using Ontology Hierarchy [22]

Tema del paper

Los autores plantean un método para la identificación automática del tema de un documento web. En su proyecto utilizaron una ontología jerárquica relacionada para poder identificar el tema del documento. Un aspecto a destacar es el modo que enriquecieron la ontología por medio de la utilización de relaciones semánticas encontradas en WordNet, las relaciones usadas fueron: sinónimos, hiperónimos/hipónimos y merónimos/holónimos. Este proceso de enriquecimiento tiene como objetivo poder tener una tasa mayor de palabras clave mapeadas y así tener una probabilidad menor de error en la identificación del tema del documento.

Las palabras clave extraídas del texto son mapeadas en el concepto correspondiente de la ontología. Usando esos conceptos relacionados, se puede obtener la relación semántica entre las palabras en el texto.

Resumen

El objetivo es la identificación de un tema principal de un documento a partir de la explotación de la estructura jerárquica de una ontología.

Para cumplir con el objetivo se encontraron con el inconveniente de poder mapear la representación del documento con los conceptos presentes en la ontología, es decir mapear las palabras clave del documento con la ontología original de partida (ontología de partida utilizada fue: “Web Yahoo”). Dado el limitado vocabulario presente en la ontología no se podría mapear directamente ésta con la mayoría de las palabras clave presente en los documentos. Para salvar este inconveniente se utilizó una base datos lingüística externa (WordNet) la cual permitió enriquecer los conceptos de la ontología. De esta forma se logró extender cada uno de los conceptos de la ontología con palabras obtenidas de WordNet.

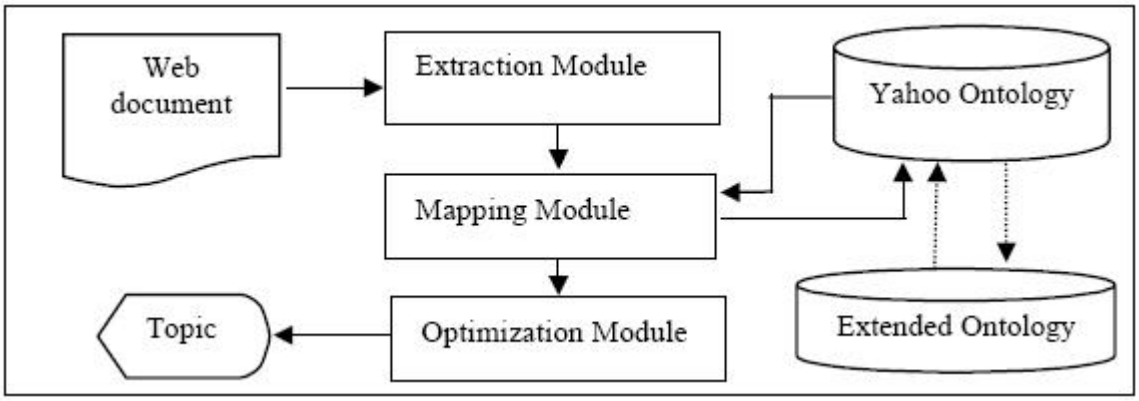


Figura A.2.: Arquitectura del sistema (ver Sección: A.1.3)

El sistema desarrollado por los autores está formado por tres componentes: El *módulo de extracción*, el *módulo de mapeo*, y el *módulo de optimización*. La entrada del sistema es un documento web y la salida es un conjunto de conceptos que refieren al tema del documento.

- *El módulo de extracción*: Es el encargado de manejar el proceso de la extracción de sentencias "importantes" del documento. Está basado en HTML tags, esto se debe a que los autores creen que algunos de los tags HTML indican la ubicación de las ideas enfatizadas por el autor del documento (por ejemplo el tag TITLE). Los autores reconocen que existen documentos que no contienen tags HTML, documentos no estructurados, éstos serán considerados en trabajos futuros. El resultado del módulo de extracción de información es una lista de palabras clave.
  Luego que se extraen todas las palabras (conceptos) destacadas del documento web, se realizan dos tareas fundamentales para poder relacionar semánticamente conceptos y poder realizar el mapeo de forma adecuada (dichas tareas probablemente sean realizadas también por nuestro software). Estas tareas son la de *"stemmed"* y *"sense-tagged"*, la primera intenta convertir las palabras clave a su lema ("forma raíz"), la segunda intenta dar un número de significado al lema obtenido. Este número es consistente con el número de significado de WordNet.

- *El módulo de mapeo*: El modulo de mapeo toma como entrada la salida del módulo de extracción. Las palabras clavé son mapeadas con los conceptos de la ontología Yahoo, la cual posee conceptos "stemmed"[5] y "sense-tagged"[6], las palabras que no se pueden mapear directamente se tratan de mapear con la ontología extendida.
  Para poder identificar los conceptos más importantes, se asocia a cada concepto un peso calculado a partir de dos variables, una es del tipo de mapeo (normal o

[5]Lema de la palabra clave.
[6]Asignación de un significado a la palabra.

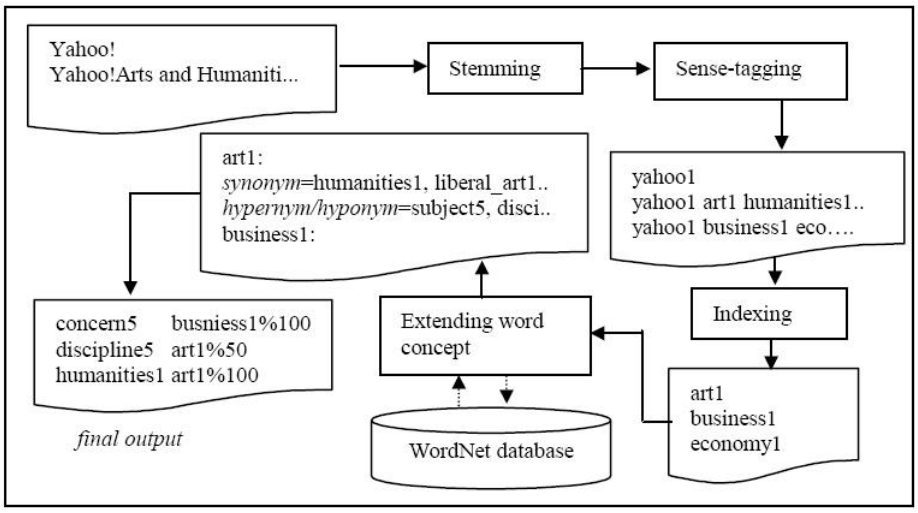


Figura A.3.: Flujo de ejecución (ver Sección A.1.3)

extendido), esto es porque al mapear en la ontología extendida la probabilidad de error aumenta dado que hubo dos procesos de *"stemmed"* y *"sense-tagged"* (uno para extender la ontología y otro para mapear la palabra clave) y la otra es la frecuencia de aparición de un concepto en el documento.

- *El módulo de optimización*: La optimización permite reducir el árbol de la ontología de forma que únicamente los conceptos activos e intermedios (entre conceptos activos) estén presente. El árbol de la ontología reducido es convertido a un único camino con los nodos más altos (ponderación mixta) utilizando el algoritmo de *Maximal Spanning Tree*, el cual se ejecuta en base los pesos "mixtos" que son asignados a los nodos como se explica a continuación[7].
  Los autores definen dos pesos para un nodo, uno *mixto* y uno *unitario*.

  - El *peso unitario* es calculado a partir del número de veces que aparece la palabra clave en el documento y la manera que ésta es mapeada con su correspondiente nodo como se explicó anteriormente. Luego que el nodo tiene su peso, éste es reducido si fue mapeado con un concepto extendido (ontología extendida) y ese concepto no fue derivado con la relación semántica sinónimo (el concepto no se formó en la ontología extendida a partir de un sinónimo en la ontología Yahoo), en este caso se reduce el la ponderación en hasta un 50 %.
  - El cálculo de la *ponderación mixta* se crea sumando todas las ponderaciones mixtas de los nodos hijos más la ponderación del nodo padre multiplicado por la cantidad de conceptos hijos + 1.

[7]Vamos a profundizar en los cálculos de pesos a los nodos porque creemos que este punto es de utilidad tenerlo en cuenta para la determinación de correspondencias semánticas de nuestro proyecto.

	Meta-Topic	*Single Path*	*TopicNode*
Precision	69.8	51.9	29.7

Cuadro A.1.: Resultados obtenidos (*Meta-Topic*: El nodo localizado un nivel por debajo del nodo raíz), sección A.1.3

Luego del cálculo anterior se elige el concepto más relevante para asignarlo como título del documento con una heurística que se explica a continuación: A cada nodo se le resta una ponderación virtual que es calculada dividiendo la ponderación del nodo raíz entre el nivel en que se encuentra el nodo (el nodo raíz se encuentra en el nivel 1). El que tenga el valor más alto (Ratio Balance) luego de esta resta es el nodo elegido como supuesto tema del documento.

Algunos de los resultados a destacar en este desarrollo es que sólo el 57.8 % de las palabras pudo ser mapeado directamente en el concepto correspondiente y un 17.8 % en los conceptos extendidos.

Para finalizar el análisis de este trabajo, en el Cuadro A.1 se presentan los resultados que se obtuvieron utilizando la fórmula:

$$Precision = \frac{hits}{hits + mistakes}$$

Conclusiones de los Autores

En este paper, los autores presentaron y evaluaron un enfoque a la identificación temática automática de un documento web a partir de la explotación de la estructura de una ontología web jerárquica (Yahoo). También mostraron como enriquecer los conceptos de la ontología web usando WordNet como base de datos de lingüística externa. La unión de los conceptos de WordNet con los de la ontología fue realizada utilizando tres tipos de relaciones semánticas existentes en WordNet (sinónimos, hiperónimos / hipónimos y merónimos/holónimos).

A.1.4. Knowledge-based Automatic Topic Identification [23]

Tema del paper

El paper presenta un método para la identificación de temas de forma automática usando ontologías jerárquicas

Resumen

Nuevamente encontramos el uso de WordNet para poder representar y relacionar semánticamente los conceptos del documento. Básicamente se plantea un método para encontrar el tema central del texto basado en identificar y ponderar los conceptos

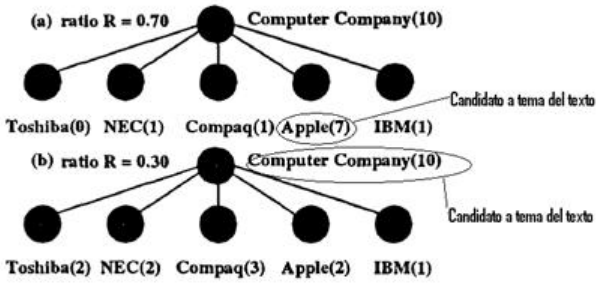


Figura A.4.: Ejemplo de cálculo de Ratio
Cuanto más cercano a 1 es el ratio el concepto tiene menos poder de generalización de los conceptos hijos, cuando es cercano a 0 el poder de atracción es adecuado para indicarlo como tema.

existentes en el documento. Para representar conceptos abstractos (generales), utilizaron la base de datos taxonómica WordNet. Además presenta el método de conteo de palabras y propone el método de conteo de conceptos.

Con el objetivo de contar la frecuencia de los conceptos, emplearon una generalización jerárquica de conceptos, construida a partir del texto (corpus) y WordNet. Utilizando la jerarquía, se enfrentaron al problema de encontrar la más apropiada generalización. Claramente no se pueden usar las hojas ya que en este nivel no se ha adquirido suficiente generalización. Tampoco se podrá utilizar el nivel más alto, nos vamos a lo muy general, consideramos que este proceso puede ser de gran valor para nosotros para determinar que tan cercana es la relación semántica entre la entrada proporcionado por el usuario de nuestro sistema y fragmentos del corpus (documento).

Como solución proponen *el ratio de frecuencia* de conceptos y *"starting depth"*.
El *peso del concepto* (frecuencia del concepto C) es a la suma de la frecuencia de los conceptos hijos más su propia frecuencia. Se puede observar que un concepto más general (el concepto padre de otro) tiene peso mayor o igual al máximo de todos los pesos de los hijos.

El ratio de cualquier concepto C lo definieron mediante la siguiente fórmula:

$$Ratio\,(C) \;=\; \frac{\text{máx}(peso\,de\,todos\,los\,hijos\,de\,C)}{\sum(peso\,de\,todos\,los\,hijos\,de\,C)}$$

R(C) es una manera de identificar el grado de síntesis del concepto (nodo). Cuanto más alto sea el ratio, menos el concepto C generaliza sobre los hijos. Por ejemplo cuando generaliza en solo un hijo, el valor de ratio es 1 por definición, es el caso del valor más alto del ratio y generaliza en un solo hijo. En este caso como tópico se tendría que elegir el hijo, dado que el concepto hijo es muy referenciado o destacado. Un concepto con un radio pequeño "referencia a sus hijos de forma más equitativa", si nos quedamos con el hijo perdemos mucha información relevante.

Con el objetivo de elegir el concepto con la apropiada generalización utilizaron el parámetro *"starting depth"* mencionado anteriormente. Se determino que el criterio de ratio sea aplicado después que se haya excedido el *"starting depth"* como profundidad en la jerarquía. La primera colección de nodos interesantes luego de pasada la profundidad será la colección seleccionada. *"Starting depth"* fue determinado experimentando con distintos valores y eligiendo el mejor.

Conclusiones de los Autores

Se llegó a buenos resultados, igualmente la *precision* rondaba en el 30 % de aciertos.

A.1.5. The Effect of Using Hierarchical Classifiers in Text Categorization [24]

Trabajo realizado por Stephen D'Alessio, Keitha Murray, Robert Schiaffino Department of Computer Science Iona College New Rochelle.

Tema del paper

Dado un conjunto de categorías, se considera el problema de asignar a un documento la categoría (del conjunto) que mejor lo represente.

En el paper se estudia la posibilidad de hacer uso de ninguna, una parte o la totalidad de la estructura jerárquica de categorías para mejorar la eficacia y la eficiencia de la categorización. Se describe un procedimiento para la generación de una jerarquía de clasificadores que modele la jerarquía de la estructura. Se presenta en una experiencia computacional de utilización de este procedimiento. Se demuestra que el uso sensato de una jerarquía puede mejorar significativamente tanto la velocidad como la eficacia del proceso de categorización.

Resumen

El problema de la categorización de documentos, consiste en asignar a un nuevo documento una o más categorías pertenecientes a un conjunto de categorías ya existente.

Como corpus de pruebas utilizaron *"The Reuters-21578"*.

Como medidas de efectividad de los algoritmos se utilizan: *precision*, *recall* y *F-measure*.

Un algoritmo para la categorización de documentos donde los pesos de los vocabularios y los términos están asociados con categorías a cada nivel de la taxonomía y donde el proceso de categorización itera sobre los distintos niveles de la jerarquía. Un término puede ser discriminado como un nivel en la taxonomía recibiendo un gran peso y transformándose en una palabra de terminación para otro nivel. El algoritmo utiliza la jerarquía para realizar la selección, reduciendo el número de términos asociados a cada categoría, resultando en un método más eficiente.

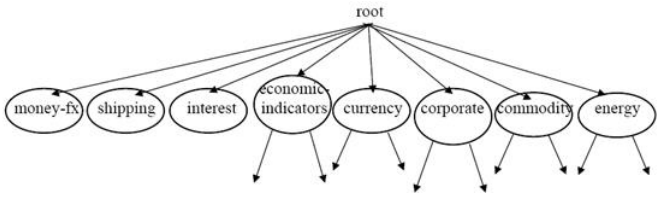


Figura A.5.: Árbol jerárquico

Los autores estudiaron métodos (conocidos como clasificadores lineales) con enfoques similares que reducen en uno o dos órdenes de magnitud sin que esto afectara la eficiencia.

Se enfocaron en un procedimiento que hace uso extensivo de las jerarquías. Esto debido a tres motivos principales: El primero es que hasta la fecha se ha demostrado que la *precision* y el *recall* disminuyen con el aumento de las categorías. Una segunda motivación es que sin una categorización establecida se pueden afrontar un número mayor de problemas, esto trae como problemas que al crecer las categorías la necesidad de un vocabulario (dominio) definido crece también. La tercer motivación se da por la introducción de un nivel de jerarquía intermedio a partir de la agregación de categorías de nivel inferior, esto hace que la decisión para algunos niveles sea la de seleccionar una categoría para cada documento, lo cual puede hacerse con gran *precision*, dado que aunque un documento pueda tener múltiples categorías, desde el punto de vista jerárquico solo tiene una.

El enfoque utilizado en el procedimiento es realizar una búsqueda Greedy por la jerarquía buscando la mejor solución para cada nivel. Se comienza con una jerarquía inicial trivial donde todas las categorías son hojas de la raíz y luego se comienza a introducir niveles intermedios de categorías.

Como se menciono antes, los autores usaron el corpus *"The Reuters-21578"* al cual lo dividieron en 70 % para desarrollo y un 30 % para validación.

El proceso básico de categorización es usado en situaciones donde tenemos un conjunto de categorías sin una jerarquización entre ellas. Se puede considerar esta como una jerarquización trivial y es modelada creando la raíz del árbol jerárquico y colocando cada categoría del conjunto como hijo de la raíz. De esta forma se arma un primer árbol jerárquico a partir de las categorías, este árbol es ponderado y a cada nodo se le agrega información del nivel y del peso.

Conclusiones de los Autores

Comenzando con una jerarquía trivial y una simple categorización binaria, se desarrollaron una serie incremental de jerarquías y categorizadores. En cada uno de los pasos se lograron mejoras sobre los sistemas de categorización trivial. Se logro una mejora en la velocidad de procesamiento en un factor de 3. A partir de los desarrollos y resultados de las pruebas se puede concluir que los sistemas de categorización pueden mejorar su performance a través de la utilización de categorías jerarquizadas.

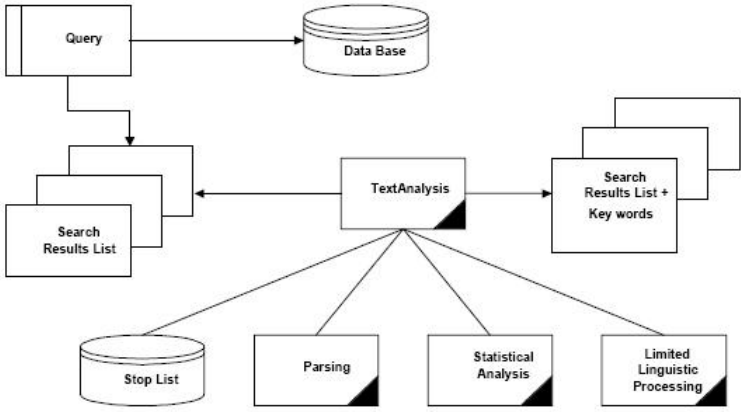


Figura A.6.: Arquitectura del sistema (ver Sección A.1.6)

A.1.6. Using Frequently Occurring Words to Identify the Subject of a Document [25]

Tema del paper

El trabajo realizado presenta un método para la determinación del tema de un documento basado en el conteo de las palabras ocurrentes en el texto.

Resumen

Drori presenta un software para determinar automáticamente la temática de un texto utilizando un mecanismo que consiste en leer el documento y utilizar un análisis estático (sin considerar relaciones semánticas) para determinar las palabras con más frecuencias en el documento. Es decir no utiliza ninguna relación semántica para inferir relaciones, simplemente cuenta ocurrencias de palabras en el texto.

Este método presenta una debilidad importante ya que el resultado está sujeto a la cantidad de palabras del texto (tamaño del documento), es decir es más efectivo con textos grandes donde se pueden encontrar más ocurrencias repetidas de una palabra. En el desarrollo se encontraron con la dificultad de que las funciones gramaticales como *"and"*, *"of"*, *"or"*, ocurren muchas veces en un texto, por lo tanto este tipo de palabras no las consideraron para catalogar el documento. Para esto el texto es analizado después que las palabras sin sentido para determinar el tema del documento son descartadas usando una lista denominada *"Stop List"* por los autores, y después de un procesamiento lingüístico básico de los elementos. Este limitado procesamiento lingüístico tiene la tarea de reconocer las inflexiones y las palabras sin sentido que serán descartadas por la lista *"Stop List"* mencionada anteriormente.

Una centena de documentos de dos géneros distintos (de distinta temática) fueros utilizados para probar y evaluar el sistema, se estudiaron resultados comparando la lista de palabras brindada por el sistema con las palabras clave y las palabras del título.

Conclusiones de los Autores

El software puede ser utilizado para identificar el tema de un documento hasta en un 71 % de los casos, utilizando las 5 palabras con mayor frecuencia inferidas por el sistema.

El resultado está sujeto a la cantidad de palabras del texto (tamaño del documento), es decir es más efectivo con textos grandes, donde se pueden encontrar más ocurrencias de una palabra.

La identificación de las palabras clave (*"keywords"*) y el título combinados tiene menos éxito que la identificación del título o las palabras clave por separado.

La identificación de las palabras clave tiene un resultado un poco más exacto que la identificación de las palabras del título, igualmente existes documentos que no contienen una lista de palabras clave, en ese caso se puede utilizar los títulos obteniendo una tasa de acierto semejante.

La tasa de identificación difiere en función del género del documento.

Nuestras Conclusiones

Al depender del tamaño del documento y no tomar en cuenta las relaciones semánticas este mecanismo no va a ser de mucha utilidad para cumplir nuestros objetivos que están fuertemente ligados a poder tener el conocimiento de lo que se está expresando en el documento y por ende tomar en cuenta la semántica de las palabras, oraciones, etc. Quizá sea de utilidad como un mecanismo de apoyo para el algoritmo principal y para identificar relevancias en el documento.

A.1.7. Identifying Topic by Position [28]

Tema del paper

Se aborda el problema de la identificación del tema de un documento a partir de la posición en el texto. Se presenta un método para encontrar los lugares más adecuados adonde probablemente estén él o los tópicos del documento, este método trabaja en base a la estructura del documento. También se presentan las pruebas y la evaluación.

Resumen

Existen múltiples tipos de métodos que buscan encontrar el tema de un documento, algunos consistían en el análisis gramatical, otros en el análisis semántico del texto, estos métodos son menos robustos para algunas entradas arbitrarias. Otros métodos, como los de *frases clave* y de *posición* son más robustos pero menos precisos. El método de posición es considerado de los mejores, éste se basa en la observación general que en la mayoría de los textos el tema principal se encuentra presente en algunas posiciones establecidas. Por ejemplo el título del documento es un muy buen lugar para buscar el tema del documento.

El método de posicionamiento tiene sus fundamentos en que los textos están basados en una estructura particular, por ejemplo, los artículos científicos (papers) tienen una sección de *abstract* en la cual se encuentra un resumen del documento. Este método no es el más adecuado en caso que la entrada varié en cuanto a estructura del documento.

Edmundson (1969) define el método de posicionamiento como: *"...the machine-readable cues are certain general characteristics of the corpus provided by the skeletons of docuements, i.e. Headings and format"*. El método de localidad está basado en las siguientes hipótesis: (1) las oraciones se encuentran bajo ciertos títulos o encabezados que están resaltados en el documento; y (2) oraciones que hacen referencia al tema tienden a ocurrir al principio o al final en el documento y de los párrafos."

Baxendale (1958) lo define como: título unión primera y última oración de cada párrafo. Notar que esta definición no es adaptable a distintos géneros de documentos, ni a distintos dominios temáticos. Además depende fuertemente del tamaño del texto. Otro inconveniente es que el tema de un documento está a un nivel de granularidad de palabra o frase, y no de un conjunto de oraciones.

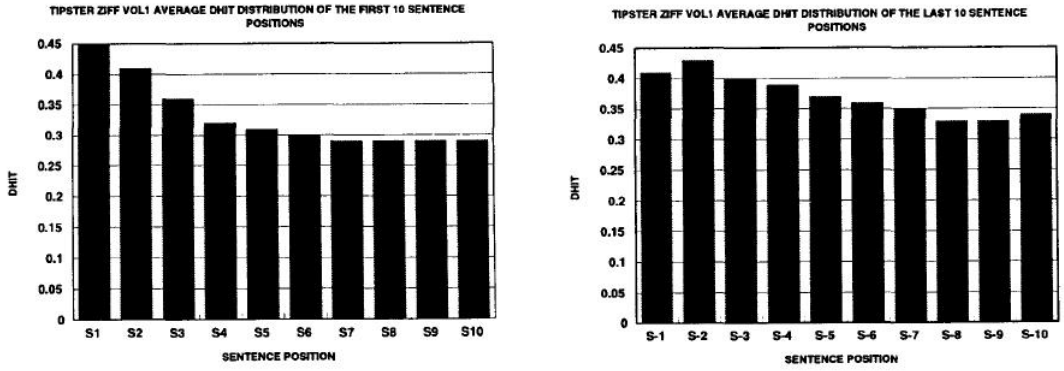
Trabajos relacionados

Edmundson's 1969 brinda las bases para el método de posicionamiento e introduce cuatro puntos clave para la identificación del tema de un documento. Entre ellas Título y Localidad (*Location*). En su trabajo Edmundson's asigna peso a las oraciones de acuerdo a su posición en el documento, dándole mayor peso a la primera oración del primer párrafo y la última del último párrafo del documento. El método planteado por Edmundson's cuenta con una *precision* de entre un 40 % y un 53 %, esta medida fue calculada comparando resultados del sistema de Edmundson con resultados seleccionados por humanos. Una de los defectos de su investigación es que nunca consideró otras posibles ubicaciones para las palabras clave, por ejemplo podrían aparecer en el segundo o penúltimo párrafo. Bexandale (1958) concluyó que en un 85 % de los párrafos el tema se encuentra en la primera oración y en un 7 % en la última. Luego Donlan (1980) tras su investigación concluyó que en los documentos de su época ya lo dicho por Bexandole no tenía gran validez dado que como resultado concluyeron que sólo un 13 % de los párrafos tienen la primera oración relacionada con el tema. Es decir el tema del párrafo puede aparecer en cualquier lugar del mismo.

Paijmans (1994) obtuvo como resultado que no hay relación entre las oraciones destacadas en un párrafo. Es decir pueden que no se den en la primera y última oración, el lugar de la oración principal o más importante puede darse aleatoriamente en cualquier ubicación.

Kieras (1985) a partir de sus estudios psicológicos confirmó la importancia de la localidad de lo que se quiere resaltar en un documento.

Dada la dicotomía entre los trabajos anteriores los autores resolvieron hacer pruebas y evaluarlas para aclarar las anteriores contradicciones y probar de manera empírica la hipótesis de que la posición de una oración en el texto refleja su importancia.



(a) Valores obtenidos para las 10 primeras oraciones de un párrafo

(b) Valores obtenidos para las 10 últimas oraciones de un párrafo.

Figura A.7.: Valores promedios relacionados con los enunciados de Baxendale's

Los autores del paper llevaron a cabo un estudio con más de 13.000 artículos[8] de diarios relacionados con anuncios de productos (concretamente del género anuncios de computación), estos artículos contaban con un conjunto de tres a ocho palabras claves y un párrafo de *abstract*.
Resumiendo para realizar las pruebas indexaron cada oración del texto como (P_n, S_m), siendo $P_n$el número de párrafo y $S_m$el número de oración. Luego para cada oración (P_n, S_m) la ponderaban con un valor igual al número de palabras clave que se mencionaban en la oración. Luego para los valores calculados en todos los artículos para un (P_n, S_m) particular calculaban su promedio, denominado *sentence yield.* Notar que el cálculo de *sentence yield* se puede extender para un conjunto de oraciones, un párrafo o un conjunto de párrafos.
Los resultados que obtuvieron fueron los siguientes:

- El título tiene una gran importancia.
- El segundo (0.75) y tercer (0.64) párrafo tienen más relevancia que el primer (0.59) párrafo.
- Lo párrafos que se encuentran en el principio del artículo tienen más importancia, tienen un contenido más informativo.
- Los párrafos que se encuentran al final no tienen gran relevancia respecto a las palabras claves.

Respecto a la hipótesis derivada de la investigación de Baxendale se concluyó lo siguiente:

- La hipótesis de que la primera oración de cada párrafo contiene información más relacionada al tópico es cierta. Esto se puede ver en la Figura A.7a.

[8]Con un promedio de 71 oraciones y 34.4 párrafos.

- La hipótesis de que la última oración también tienen una relevancia mayor también fue probada empíricamente. La Figura A.7 no refleja esto (según la figura es la penúltima) pero esto se debe a que el 47.7 % de los párrafos del corpus que utilizaron contienen solamente una oración, el 25.2 % contiene dos oraciones y el número promedio de oraciones por párrafo para el corpus es de 2.05. Por lo tanto la penúltima oración es casi siempre la primera del párrafo. Esto da como resultado que la primera oración tiene más relevancia que la última.

Los autores del paper llevaron a cabo un estudio de más de 13.000 documentos, para los cuales ya se tenían definidos los temas principales por medio de métodos manuales, y llegaron a la conclusión que las posiciones más favorables para encontrar las palabras claves son: la primer oración del párrafo y la segunda desde el final del párrafo contienen la mayoría de la información.

El análisis realizado consistió en encontrar la posición de los temas dentro de documento. Con la información obtenida para el conjunto de los documentos los autores pudieron verificar empíricamente que se cumple el método de identificación temática por posición y que los lugares más comunes para encontrar los temas principales son el primer y penúltimo párrafo.

Luego evaluaron su política de posicionamiento comparando las oraciones seleccionadas (extraídas) por el sistema y el párrafo de *abstract* generado por un humano. En esta etapa se utilizó el *abstract* y no las palabras claves como en la etapa de entrenamiento (*training*), esto se debe a que según los autores el párrafo de *abstract* provee una medida más realista e interesante de la eficacia del sistema dado que se están seleccionando oraciones del texto entero y no títulos del documento. Esta evaluación validó la hipótesis que hay ciertas posiciones en un artículo que concentran mayor información de su tópico. Nuevamente el mayor valor se dio en (P_2, S_2), decrementandose gradualmente al aumentar el párrafo y más rápidamente al aumentar la oración dentro de un párrafo.

Conclusiones de los autores

El estudio realizado por los autores, válida de forma empírica el método de posicionamiento. Además describen un método para acotar el área de búsqueda de las palabras clave en un conjunto de documentos. También llegaron a la conclusión que sólo el 30 % de las palabras clave del título o tema del documento no se mencionan explícitamente en el texto (el tema es inferido del documento) lo que hace más factible para el procesamiento automático. Calcularon que en el 50 % de los documentos las palabras clave del tema aparecen explícitamente en el título.

Nuestras Conclusiones

El paper brinda información importante de cómo están conformados los documentos y cómo se puede reducir drásticamente el área de búsqueda en la identificación del tema del documento. Aunque estos conocimientos nos brindan un aporte importante

para entender el la identificación temática consideramos que el método descripto así como los resultados de los autores no se aplican directamente al objetivo de nuestro proyecto.

En el trabajo los autores no utilizaron transformaciones morfológicas ni mecanismos de inferencia semántica, por lo cual, en caso de que se hubiera utilizado creemos que los resultados serían mejores.

Un punto débil es que utilizan las palabras clave o el párrafo de *abstract* como el tema correcto, esto hace que no pueda ser aplicable a textos con cualquier estructura.

A.1.8. Algoritmo de resolución de la anáfora pronominal en diálogos [29]

Tema del paper

Patricio Martínes-Barco, del Grupo de Investigación en Procedimiento del Lenguaje y Sistemas de Información, del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante. Presenta en su trabajo un algoritmo para la resolución de la anáfora pronominal en diálogos para el castellano. Este algoritmo combina dos aproximaciones diferentes. Por una parte, hace uso de información lingüística para aceptar o rechazar el antecedente de la anáfora mientras que por otra parte, usa información derivada de la estructura del diálogo para proponer dicho antecedente.

Resumen

Basado en un estudio de trabajos anteriores, se constato que el uso exclusivo de información de la estructura del discurso, no tienen buenos resultados. A su vez, la utilización de métodos lingüísticos clásicos sin la consideración de la información general de la estructura tampoco logra los resultados esperados.

Por esto, el autor propone la creación de un algoritmo basado en la utilización de las dos aproximaciones anteriores. Basándose en la información lingüística[9], se definen un conjunto de restricciones y basándose en la información morfológica, léxica y sintáctica[10], se define un conjunto de preferencias. Luego de obtenida está información, se define un sistema de tópicos del discurso capaz de generar información de la estructura.

Para resolver el problema el autor lo separo en tres sub sistemas. El sistema de tópicos establece los espacios de accesibilidad anafórica y proporciona los antecedentes a los sistemas siguientes. El sistema de restricciones contiene las reglas necesarias para la elección de un posible antecedente, el fallo en la verificación de una de las restricciones causa su rechazo. Por último el sistema de preferencias, permite la elección de un antecedente cuando existe más de un candidato. A su vez cada uno de

[9]Información extraída del análisis sintáctico del corpus.
[10]Información semántica basada en el uso de WordNet o EuroWordNet.

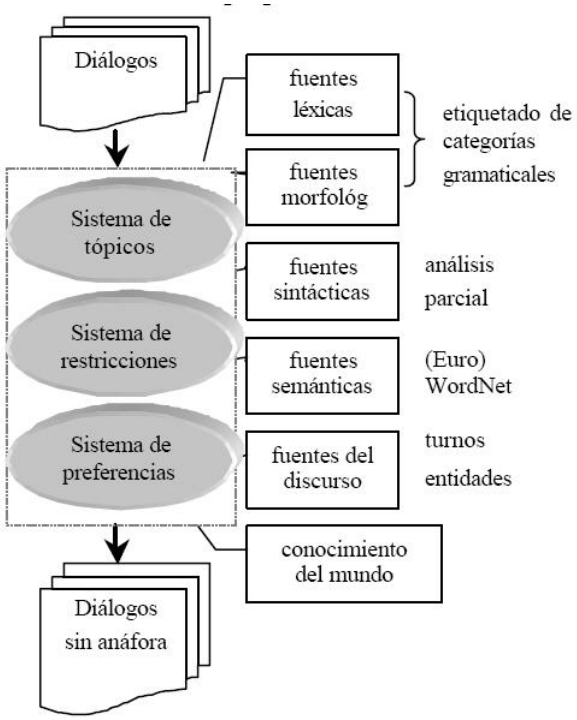


Figura A.8.: Arquitectura del sistema de resolución de anáforas

estos sistemas hace uso de varias fuentes de información, como son las fuentes léxicas, sintácticas, del discurso[11] y del conocimiento del mundo[12].

Un ejemplo del funcionamiento del algoritmo es el siguiente, dado el dialogo:

T01: <H1> Buenos días.
T02: <H2> Buenos días. ¿Qué deseas?
T03: <H1> Quiero **manzanas**.
T04: <H2> ¿De qué clase **las** quieres?
T05: <H1> No importa si son buenas.
T06: <H2> **Éstas** **las** he recibido **esta mañana**. Son muy buenas.
T07: <H1> Entonces dame de **esas**.
T08: <H2> ¿Cuántas quieres?
T09: <H1> Media docena.
T10: <H2> Muy bien, ¿qué más quieres?
T11: <H1> ¿Tienes **limones**?
T12: <H2> Sí, **los** tengo en **una caja** por aquí... Aquí están. ¿**Los** quieres muy verdes?
T13: <H1> No... Son para hacer limonada. A mis hijos **les** encanta.

[11]Esta información esta formada por una lista de entidades relevantes con sus pesos, factibles de soluciones a anáforas. Por falta de herramientas en castellano esta fuente de información fue dejada para proyectos futuros.

[12]Está fuente pretende brindarle al sistema la capacidad de simular el conocimiento del oyente/lector tiene en ciertas situaciones. Los autores dejaron su implementación para trabajos futuros.

```
Principio
  Fa :=[]; Fn :=[]; Fa' :=[]; F :=[];
  Turn :=1;
  Mientras hay turnos
    Oración :=1;
    Mientras hay oraciones
      Fn :=relevante(np(Oración));
      Fa := Fa+Fn;
      Fn :=[];
      Si Buscar_anáfora
        Buscar_antecedente(Fa)
        Buscar_antecedente(Fa')
        Buscar_antecedente(F) por peso
      FinSi
      Si Fa<>[]
        Clase(Turno):=INTERVENCIÓN;
        Si Clase(Turno-1)=INTERVENCION o
        Participante(Anterior_intervención)
           <>Participante(Turno)
          Fa' :=[];
        FinSi
      FinSi
      Oración := Oración+1;
    FinMientras
    Si Fa<>[] /* intervención */
      F :=incorporar(Fa,F);
      Fa' := Fa+Fa';
      Fa :=[];
      Anterior_intervención :=turno;
    Sino
      Clase(turno) :=CONTINUADOR;
    FinSi
    Turno :=Turno+1;
  FinMientras
Fin
```

Figura A.9.: Algoritmo de resolución de anáforas

T14: <H2> ¿Cuánto quieres?
T15: <H1> Un kilo.
T16: <H2> **Éstos** te **los** dejo a **buen precio**. Muy bien, ¿algo más?
T17: <H1> Nada más, gracias. ¿Cuánto es todo?
T18: <H2> 350. T19: <H1> Ahí tienes. Hasta luego.
T20: <H2> Hasta luego.

El algoritmo comienza considerando los turnos T01 y T02 como turnos continuadores debido a que no contienen ninguna entidad. De esta forma, las tres listas (ver Figura A.9) permanecen vacías hasta el turno T03. Al procesar T03, *manzanas* es considerado como una entidad, y por lo tanto se introduce en la lista de entidades locales actuales Fa:

$$Fa=[manzanas]$$

T03 se considera por tanto una intervención. Cuando esta intervención finaliza, las entidades de Fa se incorporan a la lista de tópicos generales F con su peso correspondiente a la vez que se guardan en la lista de entidades locales anteriores Fa'. Tras esta operación las listas quedan de la siguiente forma:

$$Fa=[\ ]\ Fa'=[manzanas]\ F=[(manzanas,10)]$$

En T04, tras la incorporación de la entidad *clase*, se detecta la primera anáfora con la aparición del pronombre *las*. En este punto el estado de las listas es:

Fa=[clase] Fa'=[manzanas] F=[(manzanas, 10)]

Para resolver la anáfora, el algoritmo busca, en primer lugar, un antecedente en la lista de entidades locales actuales (Fa). En este caso se encuentra la entidad *clase*, sin embargo, tras la intervención del sistema de restricciones se rechaza puesto que no cumple las características morfológicas requeridas (se busca un antecedente en plural). De esta forma, al no contener más candidatos la lista Fa, el algoritmo pasará a buscar en la lista de entidades locales anteriores Fa' encontrando en este caso *manzanas*. Tras la aplicación de las restricciones, *manzanas* se propone como antecedente para el pronombre y se incluye como entidad en Fa:

Fa=[clase, manzanas]

Tras procesar la primera oración de T04, Fa' se vacía:

Fa'=[ ]

Y al terminar T04 las listas quedan de la siguiente forma:

Fa=[ ] Fa'=[clase, manzanas] F=[(manzanas, 20), (clase, 10)]

Puesto que T05 es un turno continuador las listas no se modifican. Tras T06, el peso de *manzanas* se incrementa gracias a nuevas ocurrencias de esta instancia (introducidas por los pronombres), mientras que la entidad *clase* pierde peso por lo contrario.

Fa=[ ] Fa'=[manzanas, esta mañana] F=[(manzanas, 40), (esta mañana, 10), (clase, 9)]

Destacamos que las anáforas producidas por los pronombres *éstos* y *los* en el turno T16 sólo pueden ser resueltas usando las entidades de la lista de tópicos generales F puesto que las listas locales Fa y Fa' no contienen el antecedente correcto. El contenido de las listas en este momento es:

Fa=[ ] Fa'=[un kilo] F=[(manzanas,45), (limones,39), (un kilo, 10), (caja,8), (media docena,6), (esta mañana, 4), (clase,3)]

Así se cumple que, por un lado, la lista Fa está vacía, el sistema de restricciones rechaza el posible antecedente de la lista Fa', y finalmente, tanto el sistema de restricciones como el de preferencias actúan sobre la lista F devolviendo *limones* como antecedente válido.

Conclusiones de los autores

Los autores consideran que en los diálogos, a diferencia de los monólogos, es posible definir tres espacios de accesibilidad anafórica para proporcionar antecedentes a los sistemas de restricciones y preferencias. Finalmente, el algoritmo ha sido evaluado satisfactoriamente como muestran los prometedores resultados del 77 % de *precision*, teniendo en cuenta que el sistema no usa información semántica. Así mismo, han estimado que eliminando los errores introducidos por la limitación del juego de etiquetas usado, y obviando aquellos errores que se producen por arrastre de otros, podríamos alcanzar valores cercanos al 85 % y 83 % en *precision* y *recall* respectivamente.

Nuestras Conclusiones

El trabajo presentado en este paper presenta un algoritmo muy interesante y con resultados muy alentadores (según las pruebas realizadas por los autores) para la resolución de anáforas en diálogos. Algunos aspectos interesantes de destacar son la descomposición en componentes, las diversas fuentes de información (acá volvemos a encontrar la aparición de WordNet como recurso léxico), la dificultad que encontraron los autores para resolver algunos problemas y la forma en cómo fusionaron la información de la estructura del documento con la información lingüística y como esta integración permite mejorar los resultados.

A.1.9. Resolución de anáfora pronominal para el español usando el método de conocimiento limitado [30]

Tema del paper

Se presenta un método y una herramienta para resolución de anáfora pronominal para el español usando conocimiento limitado, es decir, no usar información semántica, sin embargo, se hace uso de información sintáctica.

Resumen

Dada la importancia del problema de resolución de anáfora dentro del PLN, esta tarea a sido afrontada desde distintas puntos de vistas en una variedad de sistemas de cómputo.

Existen dos tipos de métodos de resolución de anáfora:

- Basados en conocimiento adicional de varios tipos.
- Basados en conocimiento limitado (*knowledge poor*).

Existen una gran cantidad de trabajos que coinciden con la necesidad de utilizar la semántica como fuente esencial para la resolución adecuada y completa de la anáfora, aunque todavía no hacen pleno uso de ella por falta de analizadores semánticos confiables.

También se han planteado métodos de resolución enriquecidos que combinan y hacen uso de la semántica y la sintaxis, lo hacen para el inglés, en dominios restringidos con definiciones puramente manuales de jerarquías y rasgos.

Otros métodos alternativos también incorporan los papeles sintácticos en patrones de co-ocurrencia mediante estrategias puramente estadísticas (Dagan and Itai, 1991).

El trabajo presentado en este paper hace uso de un método simple basado en conocimiento limitado y puede aplicarse a muchos escenarios (por ejemplo, textos de dominio restringido o de vocabulario controlado) con resultados satisfactorios. Estos métodos de conocimiento limitado no realizan ningún tipo de análisis lingüístico complejo. Simplemente se basan en heurísticas y/o aplican algún tipo de análisis sintáctico parcial.

El sistema presentado implementa una modificación al algoritmo MARS de Mitkov y Stys (1997) propuesto para el idioma inglés, el cual propone el uso de conocimiento limitado para la resolución de análisis pronominal en manuales técnicos. Este poco conocimiento abarca una serie de reglas gramaticales correspondientes a sintagmas nominales[13], información morfológica y un conjunto de indicadores de antecedentes (*antecedent indicators*). Cada uno de estos indicadores asignará valores numéricos a los antecedentes, escogiéndose finalmente como antecedente de la expresión anafórica el que tenga mayor valor. En caso de empate se utilizan dos criterios para seleccionar el antecedente correcto. Estos dos criterios son, el antecedente que tenga mayor reiteración léxica y en caso de que persista la igualdad, se escoge el más cercano.

Para el sistema propuesto en el paper se utilizaron solo un subconjunto de los indicadores de antecedencia anteriormente mencionados, estos pueden ser vistos directamente en el paper. Es importante mencionar que los valores de ganancia para estos indicadores fueron determinados a partir de observaciones empíricas y no son considerados como óptimos o exactos.

El algoritmo propuesto presenta 5 fases:

En la primera fase, el algoritmo realiza un parseo sintáctico. Este parser devuelve información morfológica, funciones sintácticas, número gramatical y las relaciones de dependencia entre los elementos del texto que facilitan la extracción de frases nominales complejas.

En la frase 2, se identifican los pronombres anafóricos, en su implementación estos son pronombres en tercera persona y pronombres posesivos en plural y singular que demuestren anáfora nominal de identidad de referencia.

En la fase 3, para cada pronombre anafórico identificado en la fase anterior, se extraen los antecedentes potenciales (candidatos), estos se buscan desde la posición del pronombre hasta el mínimo entre 3 oraciones hacia atrás o el principio del párrafo (lo que tenga menor cantidad de texto). Una vez identificados se hacen chequeos mínimos de algunas restricciones (género, número) para que finalmente conformen el conjunto de candidatos.

[13]Ver Glosario: sintagma, sintagma nominal.

En la fase 4, se asignan valores aplicando factores preferenciales y represivos al conjunto de candidatos.

En la fase 5, el candidato con mayor valor es seleccionado como el antecedente del pronombre. En caso de empate se selecciona el antecedente más cercano al pronombre.

Conclusiones de los autores

Se desarrolló una herramienta que permite realizar la resolución de anáfora pronominal para el español modificando el método de R. Mitkov, basado en conocimiento limitado. Otra diferencia importante es que se utilizo información sintáctica.

En las pruebas realizadas con una colección de textos pequeña el método presentó una efectividad del 92,3 %.

Nuestras Conclusiones

El algoritmo presentado para la resolución de anáfora pronominal es realmente simple, está basado en un algoritmo realizado para el inglés y el polaco. La efectividad obtenida es mayor a 90 %, tanto por el sistema base como por el sistema construido para el idioma español. Es decir tiene un alto grado de *precision*.

También se destaca la importancia de contar con información semántica y principalmente sintáctica para estos métodos.

A.1.10. Using WordNet for Word Sense Disambiguation to Support Map Construction [26]

Tema del paper

Se presenta un algoritmo para desambiguar el significado de una palabra o palabras presentes en un mapa de conceptos o grafo de conceptos (*concept map*), este mapa de conceptos brinda un contexto para que a través de las relaciones de WordNet inferir el significado o sentido e la palabra ambigua perteneciente al mapa de conceptos.

Resumen

Un mapa de conceptos es una herramienta que sirve para organizar, representar y compartir conocimiento. Una de las dificultades cuando se quiere crear un mapa de conceptos es que no se sabe cuales conceptos utilizar o agregar y cuales palabras o frases describen mejor las relaciones entre estos.

El mapeo de conceptos es un proceso que crea conocimiento, dado una lista de conceptos los organiza en una representación gráfica, donde los pares de conceptos con su relación identificada por una frase forman proposiciones.

La tarea más dificultosa en la construcción es encontrar la frase que relaciona dos conceptos de la manera más precisa posible, de forma que exprese la relación existente entre dos conceptos para formar una proposición.

Para solucionar el problema de determinar el significado de una palabra polisémica utilizando el mapa de conceptos como contexto y WordNet como herramienta de ayuda los autores presentan un algoritmo que consta de seis pasos.

Comienza con la selección de las palabras o conceptos del mapa de conceptos que serán utilizados para determinar el significado de la palabra *w*. Luego para estas palabras se buscan todos los significados que tienen en WordNet, es decir para cada palabra se encuentra todos los *synsets* de WordNet. Los *synsets* son agrupados utilizando la relación de hiperónimos de WordNet de tal manera que un *synset* por palabra es permitido en cada grupo. Luego de esta etapa se tienen varios grupos como resultado, cada uno con un peso distinto, éste dependiente del número de palabras en el grupo y la distancia de las relaciones. A partir de estos pesos, el significado elegido es el *synset* para *w* que se encuentra en el grupo con mayor peso.

Etapas del algoritmo:

- **Elección de los conceptos pertenecientes al mapa de conceptos**
  La elección de las palabras del mapa de conceptos que serán utilizadas por el algoritmo como ayuda (contexto) para determinar el significado de la palabra *w* es una tarea crucial para el éxito del mismo.
  Basándose en la topología del mapa de conceptos se elige las siguientes palabras:
  - Conceptos que están a dos relaciones de distancia de *w*. Es decir conceptos que están en la misma proposición.
  - Palabras que se encuentran en la raíz del mapa de conceptos, estas palabras son una buena representación de la totalidad del mapa conceptual.
  - Palabras que pertenezcan al mismo concepto que *w*, dado que entre estas existe una fuerte relación.

- **Asignación las palabras con sus respectivos synsets (WordNet)**
  Un *synset* es un conjunto de sinónimos pertenecientes a un mismo concepto. Si la palabra está representada en WordNet pertenecerá a por lo menos un *synset*, en caso de palabras polisémicas estará asociada a más de un *synset*.

- **Creación de las secuencias de hiperónimos**
  Se construye todas las posibles secuencias de hiperónimos donde el último elemento es uno de los *synsets* del conjunto obtenido en la etapa anterior. El *synset* Ni es un hiperónimo del elemento N_{i+1}. N_0 no posee hiperónimos. En WordNet los *synsets* pueden tener más de un hiperónimo, por lo tanto se podrá crear más de una secuencia para un mismo *synset*. En este punto se tiene todas las secuencias de hiperónimos para todas las palabras participantes en el algoritmo.

- **Creación de Grupos (Cluster)**
  Definieron un cluster como una tupla (C, l, S), adonde C es una secuencia de hiperónimos, l es un entero positivo y S es un conjunto de secuencias de hiperónimos pertenecientes al cluster y para cada $s_i \epsilon S$ los primeros l *synsets* son iguales a los de C.

La creación de los grupos se realiza de la siguiente forma: para cada secuencia q generada en la etapa anterior[14], se calculan los posibles grupos tomando q como pivot. El primer grupo creado es (q, $largo\,(q)$, $\{q\}$), luego para q se crean los grupos con $largo\,(q) - 1$, $largo\,(q) - 2$, ...,*1* de forma iterativa.

- **Selección del mejor grupo**
  Para cada grupo calculado en la etapa anterior, se calcula el peso y se selecciona el cluster con mayor ponderación. Dado un cluster $H = (C, l, S)$, el peso es calculado como $\frac{1}{\sum P_i}$, siendo Pi es la cantidad de elementos (*synsets*) de s_i que difieren de C, es decir que son distintos a C.

- **Determinación del significado de *w***
  Si hay un único cluster de mayor ponderación, el último elemento (*synset*) de C es el significado de w. Si más de un cluster posee la ponderación mayor, se selecciona el *synset* que tenga más frecuencia de uso según WordNet.

Luego de hacer algunas pruebas los autores llegaron a que la tasa de acierto del método rondaba los 75 %. Para este estudio se seleccionaron mapas de conceptos bien definidos y para la palabra seleccionada de cada mapa del concepto se verificó que existiera el significado correcto y más de un significado[15]. Estas pruebas fueron realizadas con mapas de conceptos que tenían un promedio de 22.75 conceptos, con una desviación de 9.91 y el promedio de conceptos utilizados por el algoritmo fue 9.37, con una desviación de 4.15.

La correcta determinación de los conceptos a utilizar, es decir los conceptos que sirven como ayuda contextual para poder determinar el significado de la palabra w es crucial para que con el algoritmo se obtenga un resultado correcto. En el caso que los vecinos de la palabra w no pertenezcan al contexto del concepto w (es decir un mapa conceptual mal formado o con poca información) el algoritmo puede ser poco eficaz. En estos casos una solución planteada por los autores es retornar el significado que tenga la mayor frecuencia de uso según WordNet cuando los clusters están por debajo de un umbral.

Conclusiones de los autores

Se presentó un algoritmo que tiene la capacidad de determinar el significado de una palabra perteneciente a un determinado contexto utilizando básicamente relaciones de hiperónimos presentes en WordNet.

La eficacia del algoritmo ronda los 75 % bajo algunas hipótesis:

- El mapa conceptual en el cual se encuentra la palabra o el concepto w tiene que estar bien formado.

- WordNet debe tener dos o más *synsets* asociados a la palabra w.

[14] Es decir q tiene como último elemento un *synset* s que contiene la palabra w como sinónimo.

[15] Es decir se verifico que existiera más de un *synset* en WordNet para la palabra y que uno de estos lo describa en el contexto del mapa de concepto.

A.1.11. RicoTerm 2 [19]

Tema del proyecto

En este proyecto de investigación se propone desarrollar descripciones eficientes del discurso y de la terminología de la economía, en castellano, catalán, gallego, euskera e inglés, con el objetivo aplicado de crear recursos lingüísticos multilingües, que puedan ser aprovechados por diversas técnicas de RI, y en especial por los motores de búsqueda en Internet. El equipo de investigación ya dispone de un corpus textual procesado lingüísticamente para tres lenguas (castellano, catalán e inglés), y se prevé desarrollar en este proyecto los corpus textuales complementarios del gallego y del euskera, cuya explotación permitirá diseñar estrategias generalizables para la RI. La explotación de estos corpus permitirá desarrollar otras aplicaciones, fundamentalmente de carácter semántico y fraseológico, que puedan aprovecharse en la RI: enriquecimiento de los diccionarios de procesamiento con información semántica y fraseológica, desarrollo de una ontología para el ámbito económico vinculada a una base de datos terminológica multilingüe o la adaptación de un extractor automático de terminología para el ámbito económico. Además de estos recursos, que podrán ser también utilizados en técnicas de extracción de información, los resultados esperados de este proyecto para la RI se basan en el diseño de un sistema automático de reelaboración de consultas multilingüe como *input* para los motores de búsqueda existentes. Este sistema de reelaboración de consultas usará la información de la ontología y de la base de datos terminológica para transformar una consulta simple, y ambigua, en una consulta compleja que mejore la relevancia de la respuesta dentro del ámbito de la economía.

Resumen

Los experimentos en expansión de consultas se basan en métodos que permiten detectar, a partir de la consulta de un usuario, aquello que constituye el núcleo de la consulta. En este proyecto se propuso abordar el diseño del re-elaborador de consultas sobre economía a partir de dos métodos principales: la expansión a partir de términos y la expansión a partir de textos.

- Expansión de consultas a partir de textos: Los sistemas basados en la expansión de consultas a partir de textos suelen presentar una *precision* del entorno de un 40 %(Strzalkowski et al. 1999:136), resultados demasiado bajos para usuarios profesionales. Las soluciones propuestas para mejorar los resultados se basan fundamentalmente en criterios de restricción sobre los textos en la expansión: restricción estructural y temática.

En el proyecto RICOTERM-2 se utilizaron dos estrategias textuales complementarias para la expansión de consultas: el uso de una herramienta de extracción automática de términos y el uso de un detector de relaciones conceptuales, que permitan identificarlos y etiquetarlos en el contexto. Sumado al uso de una base de datos terminológica y de diccionarios especializados sobre economía, que incluyan definiciones.

- Expansión de consultas a partir de ontologías: Los sistemas de expansión de consultas que, en los últimos años, están mejorando ostensiblemente sus resultados

son los que interactúan con ontologías o jerarquías léxicas. Los sistemas de expansión de consultas que utilizan ontologías, o alternativamente tesauros, se basan en el criterio de expansión de términos o expansión léxica, es decir que a partir de un término identificado como relevante en un documento se establecen correlaciones con conceptos u otras unidades léxicas que representan estos conceptos o conceptos afines (en otras palabras, se expande el término con otros de significado similar o de temas afines). Uno de los recursos más utilizados como ontología en sistemas de recuperación de información con expansión de consultas es WordNet. De esta forma los sistemas de expansión de consultas asocian automáticamente conjuntos de sinónimos para cada vector de consulta. Uno de los principales problemas detectados en este tipo de sistemas es que, al tratarse de un recurso de carácter general, no restringido temáticamente, las consultas simples suelen estar sujetas a ambigüedad. Estas dificultades pueden reducirse utilizando ontologías especializadas.

Conclusiones de los Autores

El diseño del sistema se corresponde con una expansión de consultas multilingüe que combina distintas técnicas. Además de métodos matemáticos, se utilizaron técnicas lingüísticas como la expansión de base léxica y la expansión mediante corpus. Consideran que utilizar un banco de conocimiento modular especializado en el ámbito de la economía con diccionarios y ontologías afines ofrece buenos resultados.

Nuestras Conclusiones

El paper presenta una idea de cómo construir un sistema de expansión de consultas.

Si bien los autores recomiendan el uso de ontologías especificas, esto no sería posible en nuestro proyecto (ya que se requiere que el dominio no esté acotado), en consecuencia se usará la ontología genérica que ofrece WordNet.

A.2. Herramientas

Se presentaran las distintas herramientas que se evaluaron durante el desarrollo del proyecto. Algunas de ellas fueron utilizadas en trabajos de investigación ya realizados y presentados en la sección A.1.

Dentro de las herramientas existentes para el procesamiento de textos en idioma español podemos destacar las siguientes:

A.2.1. WordNet

WordNet[2] es una enorme base de datos léxica en para el idioma inglés. Agrupa palabras de una categoría gramatical en un conjunto de sinónimos llamados *"Synset"*, cada *synset* representa un concepto distinto. Cada *synset* contiene una definición para el concepto y son interconectados de manera de representar las relaciones semánticas

y léxicas. Cabe aclarar que las categorías gramaticales con las cuales trabaja WordNet son nombre, verbo, adjetivo, adverbio.

La base de datos y las herramientas se han liberado bajo una licencia BSD y pueden ser descargadas y usadas libremente, lamentablemente no podemos decir lo mismo de las versiones de WordNet desarrolladas para el idioma español.

WordNet fue creado y es mantenido por el Cognitive Science Laboratory de la Universidad de Princeton bajo la dirección del profesor de psicología George A. Miller[16]. Su desarrollo comenzó en 1985.

A.2.2. EuroWordNet

EuroWordNet [3] es una base de datos multilingüe que consta de varios WordNet's para algunos idiomas europeos (holandés, italiano, español, alemán, francés, checo y estonio). Cada uno de estos WordNet's está estructurado de la misma forma que el WordNet desarrollado en Princeton (ver A.2.1). EuroWordNet estructura los WordNet de forma que queden conectados mediante el Índice Inter-Lingual(ILI). Estas conexiones permiten conectar palabras en un lenguaje con palabras similares en otros lenguajes y acceder a una ontología compartida compuesta (Top-Ontology (ver A.10)) por distinciones semánticas. Esta ontología proporciona una categorización común para todos los idiomas, mientras las distinciones específicas de cada idioma quedan en cada WordNet.

EuroWordNet es bastante útil para traductores y para relacionar conceptos en diferentes idiomas. Un aspecto negativo es que no está disponible para utilizarlo de forma gratuita. En contraparte WordNet de Princeton está disponible de manera gratuita, esta característica se refleja fuertemente en la gran cantidad de trabajos de investigación académica realizados en idioma inglés y la escasa cantidad de trabajos realizados para otros idiomas.

A.2.3. Spanish WordNet

Spanish WordNet [4] es una versión de WordNet para el idioma español, fue desarrollada y es mantenida principalmente por el Grupo de investigación de Procesamiento de lenguaje natural de la Universidad Politécnica de Cataluña[17]. Sigue la especificación de EuroWordNet y tiene practica-mente la misma arquitectura que la versión WordNet desarrollada en la Universidad de Princeton. La arquitectura está compuesta por un conjunto de conceptos (*synsets*) los cuales están relacionados semánticamente y constan de un conjunto un conjunto de sinónimos (*variants*). Cada concepto contiene una definición que es la definición que le corresponde al conjunto de sinónimos que agrupa, también consta de un atributo el cual lo clasifica en alguna categoría gramatical.

Una de las principales ventajas de esta versión de WordNet es la posibilidad de obtener licencias libres para su uso con fines de investigación académica.

[16] Ver http://wordnet.princeton.edu/~geo/
[17] http://www.lsi.upc.edu/~nlp/web/

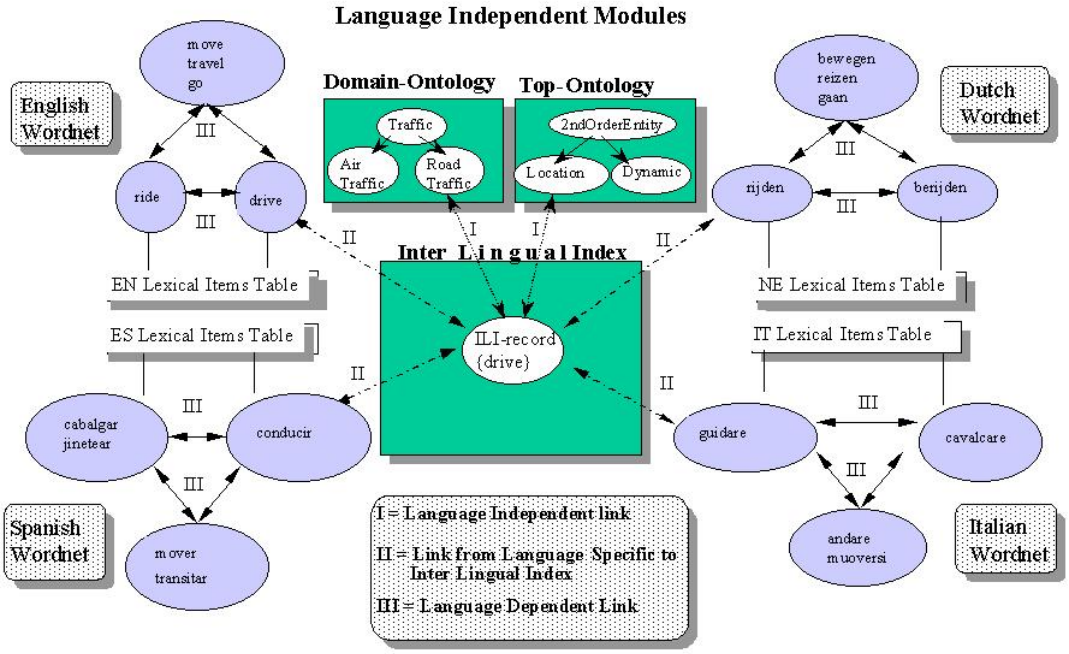


Figura A.10.: Arquitectura de EuroWordNet

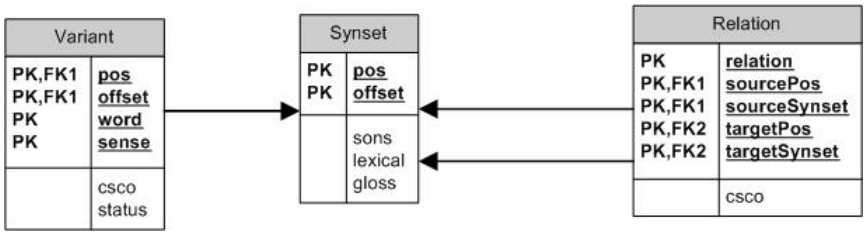


Figura A.11.: Diseño de base la base de datos Spanish WordNet

A.2.4. Freeling

FreeLing 2.0 es un programa informático gratuito de análisis lingüístico. Está desarrollado por el Centre de Tecnologies i Aplicacions del Llenguatge i la Parla (TALP) de la Universitat Politècnica de Catalunya (UPC). Permite múltiples funciones, como división en oraciones, lematización, etc., para español, catalán, gallego, italiano e inglés (posee diccionarios específicos para cada lengua), y funciona bajo sistema operativo Linux.

A.2.5. UIMA

UIMA es un software que se utiliza para analizar grandes volúmenes de datos no estructurados con el fin de obtener información relevante al usuario. Un ejemplo de un sistema basado en UIMA es el sistema resultado de este proyecto.

Apache UIMA [5]

UIMA es una plataforma escalable y extensible para crear, integrar y desarrollar soluciones que manejan información no estructurada a partir de la combinación de módulos. Permite que las aplicaciones UIMA se descompongan en componentes, de esta forma logrando soluciones escalables y reutilizables, cada componente realiza una tarea particular y tiene como entrada un documento y posibles salidas de módulos anteriores.

Es importante mencionar que Apache UIMA es una implementación *open source* de una especificación de UIMA[18]. Proporciona un framework el cual facilita el desarrollo de software UIMA. Para desarrollar aplicaciones UIMA con la especificación de apache hay que implementar interfaces brindadas y definir ciertos archivos de metadatos. El framework maneja los componentes y el flujo compuesto por los mismos.

A.2.6. Lavinia

Lavinia es un ambiente basado en UIMA para Procesamiento de Lenguaje Natural, desarrollado por el Grupo de Procesamiento de Lenguaje Natural[19] del Instituto de Computación, Facultad de Ingeniería, Universidad de la República en Montevideo, Uruguay.

Sus principales características son:

- Es un ambiente web, por lo que puede ser utilizado sólo con un navegador.
- Utiliza la arquitectura Apache UIMA como motor de procesamiento y como formato de intercambio de la información entre los diferentes módulos.

[18] Especificación UIMA
[19] Ver [17].

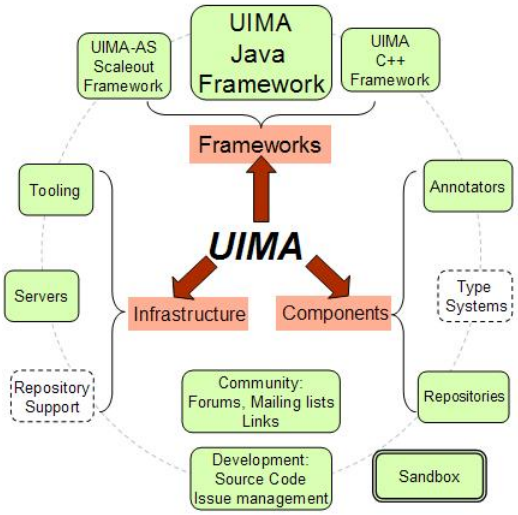


Figura A.12.: Apache -UIMA : Frameworks, Components, Infrastructure

- Provee mecanismos de fácil uso para el armado de flujos de análisis de textos, incluyendo la capacidad de parametrización y especificación de recursos externos.
- Provee un mecanismo homogéneo y amigable de visualización de las anotaciones sobre los textos analizados, incluyendo anotaciones solapadas.
- Permite exportar los resultados en formato XML, e importar resultados de análisis anteriores.
- Cualquier módulo desarrollado para la plataforma UIMA puede incorporarse rápidamente a Lavinia.
- Incluye varios componentes ya implementados para realizar tareas comunes de procesamiento de Lenguaje Natural.

A.3. Conceptos

Homonimia

La homonimia es una relación léxica que se establece entre dos homónimos, es decir, palabras que presentan identidad formal (fónica o gráfica) pero diferencia en el significado. Es la cualidad que se da en una lengua o idioma, cuando existen palabras que tienen un significado diferente, aunque se escriben o pronuncian igual. Las palabras homónimas pueden ser homógrafas cuando tienen una escritura igual u homófonas cuando poseen la misma pronunciación.

La palabra *"lista"* es homógrafa ya que puede referirse a una mujer inteligente, o bien a un conjunto de elementos.

Ejemplo de palabra homófona, pueden ser las palabras *"hola"* y *"ola"* que claramente se escriben diferentes pero se pronuncian igual.

Polisemia

Se denomina polisemia a la capacidad que tiene una sola palabra para expresar distintos significados. Pluralidad de significados de una palabra o de cualquier signo lingüístico, con independencia de la naturaleza de los signos que lo constituyen. Al igual que la homonimia, en el caso de la polisemia se asignan varios significados a un solo significante. Pero, mientras la homonimia se produce por coincidencia de los significantes de diversos signos, la polisemia se debe a la extensión del significado de un solo significante.

La palabra *"sierra"* es polisémica ya que puede usarse para referirse a una herramienta o a una cordillera.

Inflexión Gramatical

La flexión es la alteración que experimentan las palabras mediante morfemas constituyentes según el significado gramatical o categoría para expresar sus distintas funciones dentro de la oración y sus relaciones de dependencia o de concordancia con otras palabras o elementos oracionales.

En gramática tradicional, la flexión suele recibir nombres diferentes según se aplique a diferentes clases de palabras, la flexión verbal suele denominarse conjugación, la flexión nominal suele denominarse declinación, y en lenguas indoeuropeas se aplica normalmente a sustantivos, pronombres y adjetivos.

La flexión nominal del español tiene como morfemas[20] el género (masculino, femenino y neutro) y el número (singular y plural), entre otros.

En la flexión verbal tiene como morfemas el modo (indicativo, subjuntivo e imperativo), el tiempo (presente, pretérito y futuro), el número (singular y plural) y la persona (primera, segunda y tercera).

Lema

Abstracción, a partir del haz de rasgos flexivos de una palabra, que representa a esta como forma canónica. Un lema representa un conjunto de palabras con la misma raíz, misma categoría léxica (tipo de palabra) y mismo significado. Normalmente los lemas coinciden con las entradas de un diccionario.

[20] El morfema constituye la parte variable de la palabra. La palabra tiene dos tipos de monemas: lexemas y morfemas (gramaticales). El morfema, con valor gramatical, aparece siempre asociado al lexema, con valor semántico. Por ejemplo para la palabra *gatos* está formado por el lexema *gat* y adquiere significados más específicos sobre el género y número con los morfemas flexivos *-o* (masculino) y *-s* (plural).

- El lema de los adjetivos calificativos será siempre la forma masculina singular *"bonito"* o la forma singular si el adjetivo es de género común *"alegre"*. Paras los adjetivos invariables, es decir aquellos que tanto para el singular como para el plural presentan la misma forma, el lema y la forma han de coincidir.
- El lema de los adverbios acabados en -mente es la misma forma a adverbial acabada en -mente, es decir, el lema de *"rápidamente"* es *"rápidamente"*
- Los nombre tienen como lema la forma singular, tanto si es de género femenino como masculino o neutro. Para los nombres invariables, es decir, aquellos que tanto para el singular como para el plural presentan la misma forma *"tesis"*, el lema y la forma coincidirán.
- El lema de un verbo ha de ser siempre el infinitivo.
- Para los pronombres el lema será la forma masculina del pronombre con las mismas características de caso y persona.
- El lema de los signos de puntuación es el propio signo.
- El lema de las cifras y numerales es el valor de estos, por ejemplo *"239"* tiene lema *239* y *"doscientos veinte"* tiene lema *220*.

Familia léxica

Una familia léxica es el conjunto de palabras que comparte un mismo lexema. Una palabra, llamada prima, acarrea con la mayor información semántica o de contenido, mientras que las demás, formadas por derivación (es decir, añadiendo afijos), aportan matices de significado. Algunos ejemplos: Familia léxica de hierba (herb-, del latín herba) Lexema hierb-. Hierba/(yerba) hierbabuena hierbajo hierbezuela herbáceo herbajar herbaje herbajear herbajero herbar herbario herbaza herbazal herbecer herbero herbicida herbívoro herbolar herbolario herbolecer herboristería herborización herborizador herborizar herboso.

B. Diseño

En este anexo se pretende continuar la descripción de la arquitectura, detallando en profundidad el diseño del sistema, explicando en detalle cada uno de los componentes, sus funcionalidades, métodos, interfaces que implementan e interacciones entre ellos.

B.1. Diseño del componente Freeling2

El núcleo del componente es la clase *Freeling2Annotation*, esta clase extiende a *JCasAnnotator_ImplBase* lo que permite la integración con UIMA (ver Figura B.1). Cada componente UIMA "primitivo" (los otros componentes existentes son los *"agregate"* que tienen el objetivo de determinar flujos de procesamiento y no brindar funcionalidad) contiene una clase que extiende de *JCasAnnotator_ImplBase,* ésta contiene el método *process(JCas aJCas)* que será invocado por el framework UIMA cuando corresponda. La clase *uima.jcas.tcas.Annotation* brinda funcionalidad y comportamiento para poder marcar segmentos o partes del documento, las clases *Nombre*, *Verbo*, *Adjetivo*, *Adverbio*, *Pronombre*, *Sentencia* la extienden agregándole nuevos atributos. Estas clases podrán ser recuperadas y utilizadas por componentes siguientes en el flujo de procesamiento de UIMA. La tarea de serializar y deserializar los objetos pertenecientes a estas clases para su pasaje entre los distintos componentes la realiza el UIMA, brindando al programador la sensación de estar en el mismo componente.

El objetivo principal del componente es realizar un análisis morfológico del texto. También determinar para los verbos, adverbios, adjetivos y nombres sus posibles significados (*offsets*). Otra funcionalidad que realiza, es la identificación de las oraciones presentes en el documento. Para llevar a cabo estas tareas se interactúa con el componente de Utilidades, de esta forma se invoca Freeling por medio de la clase *RunCommandFreeling*.

RunCommandFreeling es la encargada de invocar el ejecutable de Freeling, capturar el resultado y generar los *data types* que representan las oraciones y la información morfológica que luego van a ser anotados por el llamador (en este caso la clase *Freeling2Annotation*).

B.2. Diseño del componente Identificador de Relaciones

El componente *identificación de relaciones* (analizador de relaciones) es el encargado de encontrar relaciones semánticas entre las palabras del documento y de la

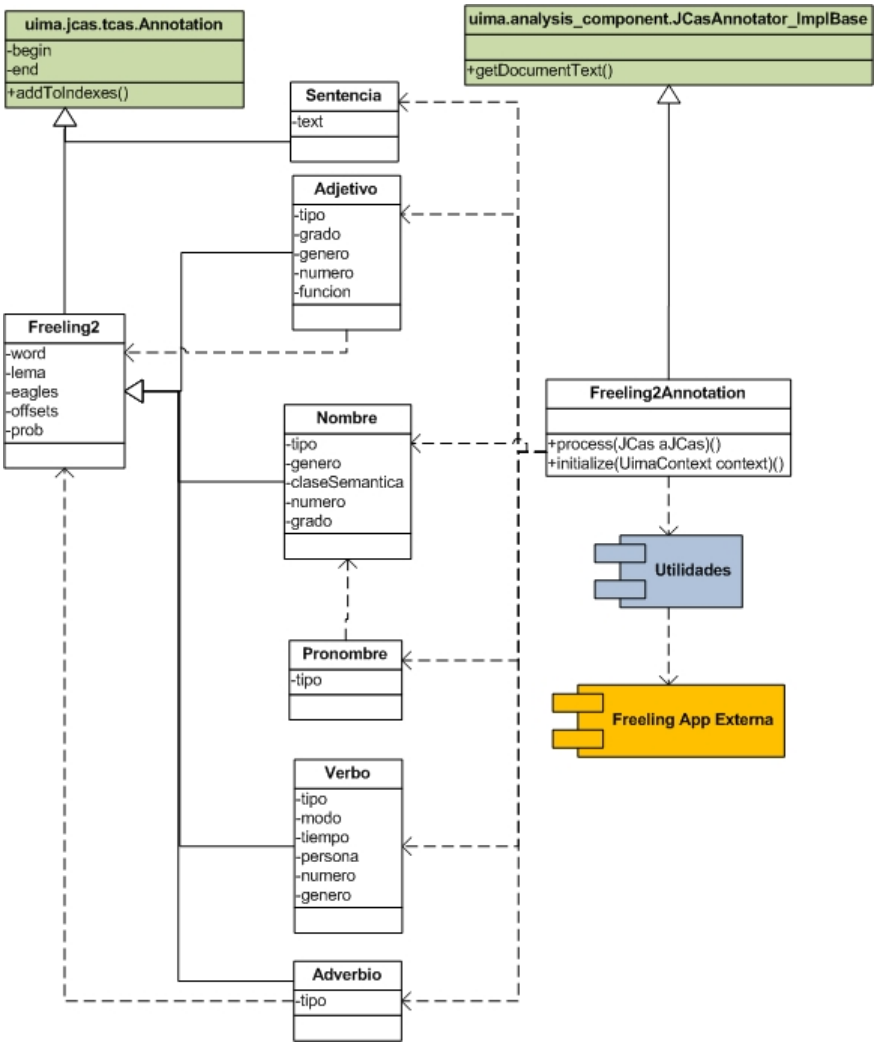


Figura B.1.: Diagrama de diseño del componente Freeling2

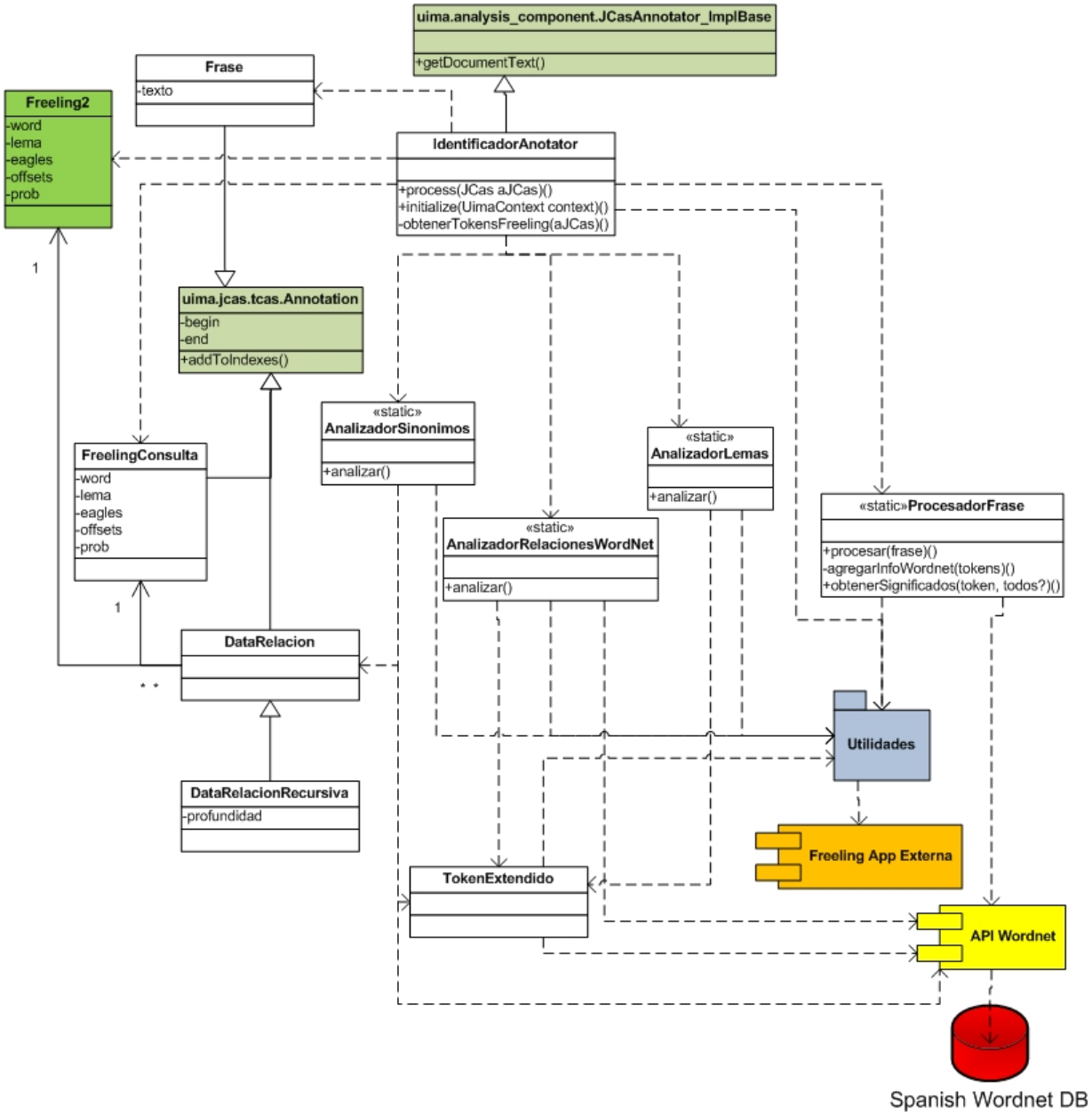


Figura B.2.: Diagrama de diseño del componente Identificador de Relaciones

Algorithm B.1 Código para la obtener los nombres, verbos, adjetivos y adverbios.

```
private Collection<Freeling2> obtenerTokensFreeling(JCas aJCas)
{
    Collection<Freeling2> tokens = new ArrayList<Freeling2>();
    FSIterator it = aJCas.getAnnotationIndex().iterator();
    FSTypeConstraint constraint = aJCas.getConstraintFactory().createTypeConstraint();
    constraint.add((new Nombre(aJCas)).getType());
    constraint.add((new Verbo(aJCas)).getType());
    constraint.add((new Adverbio(aJCas)).getType());
    constraint.add((new Adjetivo(aJCas)).getType());
    it = aJCas.createFilteredIterator(it, constraint);
    while (it.hasNext())
    {
        Freeling2 fa = (Freeling2)it.next();
        tokens.add(fa);
    }
    return tokens;
}
```

frase de consulta y por lo tanto es un componente clave dentro del sistema. Utiliza los resultados del componente anterior para cumplir con su objetivo.

Luego de la obtención de los *annotators* indexados por el componente anterior (ver Algoritmo B.1), se procede a analizar la frase de consulta de forma similar a la realizada en el componente Freeling2, pero a diferencia de éste se permite seleccionar el método de desambiguación (significado más frecuente, todos los significados, o selección manual). *ProcesadorFrase* es la clase encargada de realizar la tarea anterior comunicándose con el componente de Utilidades para obtener el resultado de Freeling para la frase de consulta (ver Figura B.2).

A continuación se anotan todas las relaciones entre las palabras de la frase de consulta y las palabras del documento, las clases utilizadas para representar y marcar las relaciones semánticas extienden *DataRelacion*. Para mejorar la cantidad de relaciones posibles de identificar, se realiza una extensión de las palabras de la frase de consulta mediante la utilización del componente WordNet.

La implementación del componente utiliza la clase *IdentificadorAnnotator* que extiende *JCasAnnotator_ImplBase* permitiendo la interacción con UIMA. Esta clase es la encargada de obtener la información generada por el componente anterior (nombres, verbos, adjetivos, adverbios)[1] para luego determinar (anotar) las relaciones existentes colaborando con otras clases internas y con el componente API WordNet.

Para el reconocimiento de las relaciones, se utilizan 3 clases estáticas: *AnalizadorSinonimos*, *AnalizadorRelacionesWordnet* y *AnalizadorLemas*. Como el nombre lo indica, *AnalizadorSinonimos* (ver pseudocódigo en Algoritmo B.3) reconoce los sinónimos presentes, *AnalizadorRelacionesWordNet* (ver pseudocódigo en Algoritmo B.4) reconoce relaciones recursivas o no y *AnalizadorLemas* (ver pseudocódigo en Algoritmo B.2) determina los lemas coincidentes entre la frase de consulta y el documento.

Para el diseño del algoritmo se siguió el patrón UIMA recomendado, y los patrones de diseño *Singleton* y *Composite*.

[1] Categorías gramaticales presentes en Spanish WordNet.

Algorithm B.2 Pseudocódigo del Analizador de Lemas

```
para cada freelingConsulta en collección(freelingConsulta)
{
    Si no esFiltradaCatGtical(freelingConsulta.obtenerEAGLES()) // StopList
    {
        para cada freelingDoc en collección(freelingDoc)
        {
            Si (catGtical(freelingConsulta) == catGtical(freelingDoc)
            && lema(freelingConsulta) == lema(freelingDoc)
            {
                lemaAnnotation = crear(LemaAnnotation).
                lemaAnnotation.Begin = freelingDoc.Begin
                lemaAnnotation.End = freelingDoc.End
                lemaAnnotation.SetFreelingConsulta(freelingConsulta)
                lemaAnnotation.SetFreelingDocumento(freelingDoc)
                lemaAnnotation.AddToIndex()
            }
        }
    }
}
```

Algorithm B.3 Pseudocódigo del Analizador de Sinónimos

```
para cada freelingConsulta en collección(freelingConsulta)
{
    para cada synsetCosulta en freelingConsulta.obtenerSysnsets()
    {
        para cada freelingDoc en collección(freelingDoc)
        {
            Si (catGtical(freelingConsulta) == catGtical(freelingDoc))
            {
                para cada synsetDoc en freelingDoc.obtenerSynsets()
                {
                    if (synsetConsulta == synsetDoc)
                    {
                        sinonimoAnnotation = crear(SinonimoAnnotation).
                        sinonimoAnnotation.Begin = freelingDoc.Begin
                        sinonimoAnnotation.End = freelingDoc.End
                        sinonimoAnnotation.SetFreelingConsulta(freelingConsulta)
                        sinonimoAnnotation.SetFreelingDocumento(freelingDoc)
                        sinonimoAnnotation.AddToIndex()
                    }
                }
            }
        }
    }
}
```

Algorithm B.4 Pseudocódigo del Analizador de Relaciones

```
para cada freelingConsulta en collección(freelingConsulta)
{
    para cada synsetCosulta en freelingConsulta.obtenerSysnsets()
    {
        relaciones = synsetConsulta.obtenerRelaciones(nombreRelacion, maxProfundidad)
        para cada relacion en relaciones
        {
            synsetDestino = relacion.obtenerSysnsetDestino()
            para cada freelingDoc en collección(freelingDoc)
            {
                para cada synsetDoc en freelingDoc.obtenerSynsets()
                {
                    if (synsetDestino == synsetDoc)
                    {
                        relationAnnotation = crear(RelationAnnotation).
                        relationAnnotation.Begin = freelingDoc.Begin
                        relationAnnotation.End = freelingDoc.End
                        relationAnnotation.SetFreelingConsulta(freelingConsulta)
                        relationAnnotation.SetFreelingDocumento(freelingDoc)
                        relationAnnotation.SetProfundidad(relacion.obtenerProfundidad)
                        relationAnnotation.AddToIndex()
                    }
                }
            }
        }
    }
}
```

B.3. Diseño del componente Algoritmo de Identificación

La clase principal del componente es *AlgoritmoAnnotator* que extiende de *JCasAnnotator_ImplBase* lo que permite la integración con UIMA.

Una vez recopilada la información (tarea realizada por los componentes antes descriptos), es necesario recuperar los segmentos relacionados, lo cual puede realizarse mediante diferentes algoritmos. Estos algoritmos aunque similares en sus objetivos pueden tener diferentes resultados, y es deseable que el usuario sea capaz de seleccionar y poder crear su propio algoritmo para ser incluido en el sistema. Con este objetivo fue diseñado el componente encapsulando la mayor parte de la funcionalidad. También es posible configurar parámetros de los algoritmos presentes sin la necesidad de recompilar el sistema. De esta forma, cambiando variables que determinan su comportamiento se pueden customizar y parametrizar para mejorar los resultados según las necesidades del usuario.

Actualmente el sistema provee 3 algoritmos (criterios de selección) que asignan diferentes ponderaciones a las oraciones y seleccionan un subconjunto de ellas como solución.

En la Figura B.3 se puede ver como los tres resultados (incluyendo un 4 del usuario) se representan mediante los *annotators Resultado1*, *Resultado2*, *Resultado3* y *Resultado4* respectivamente, los cuales heredan del *annotator Resultado*.

SentenciaCompletitud es utilizada para almacenar información estadística de cada oración, de la misma forma *InfoDoc* lo hace para la frase de consulta y para el docu-

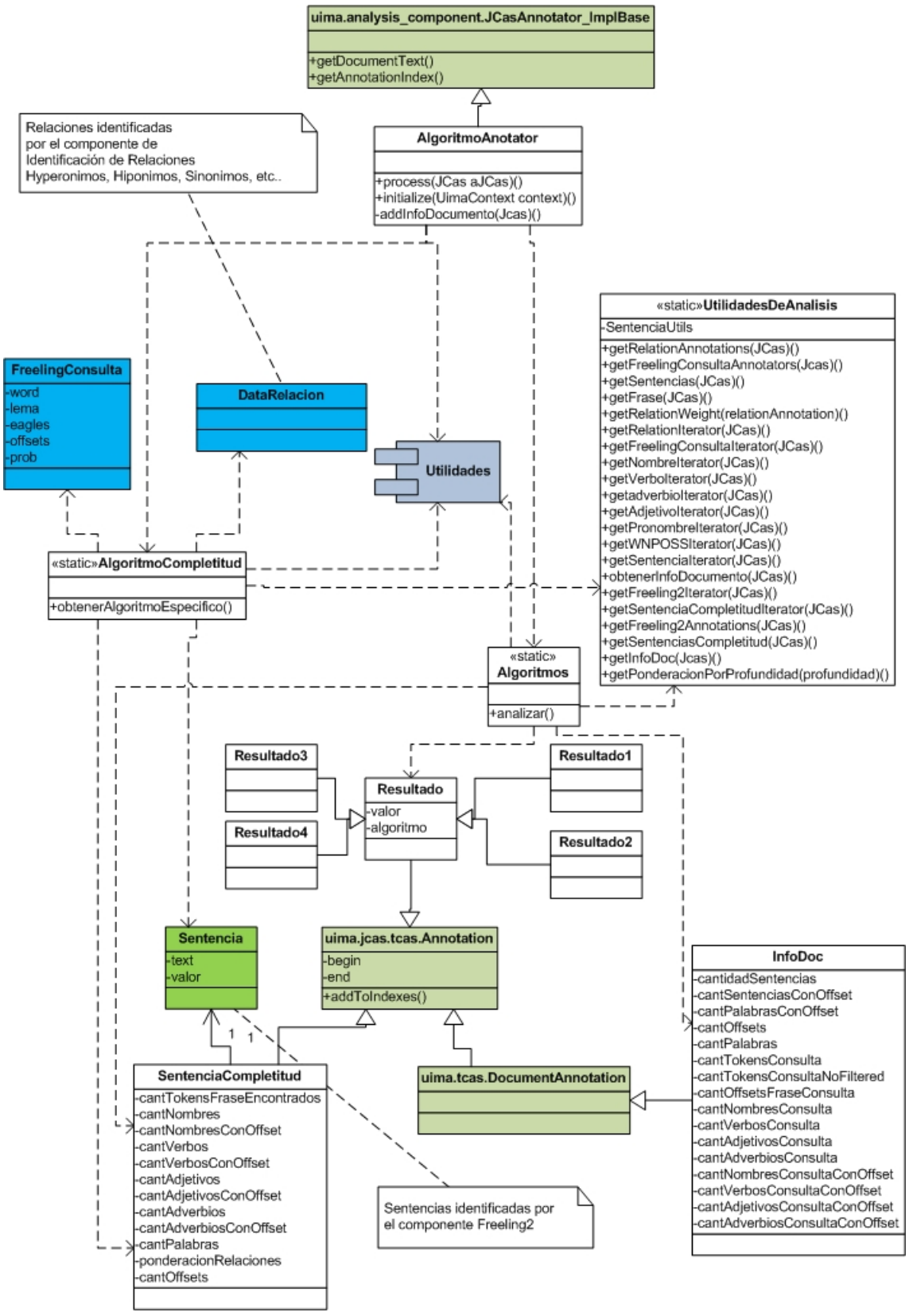


Figura B.3.: Diagrama de diseño del Algoritmo de Identificación

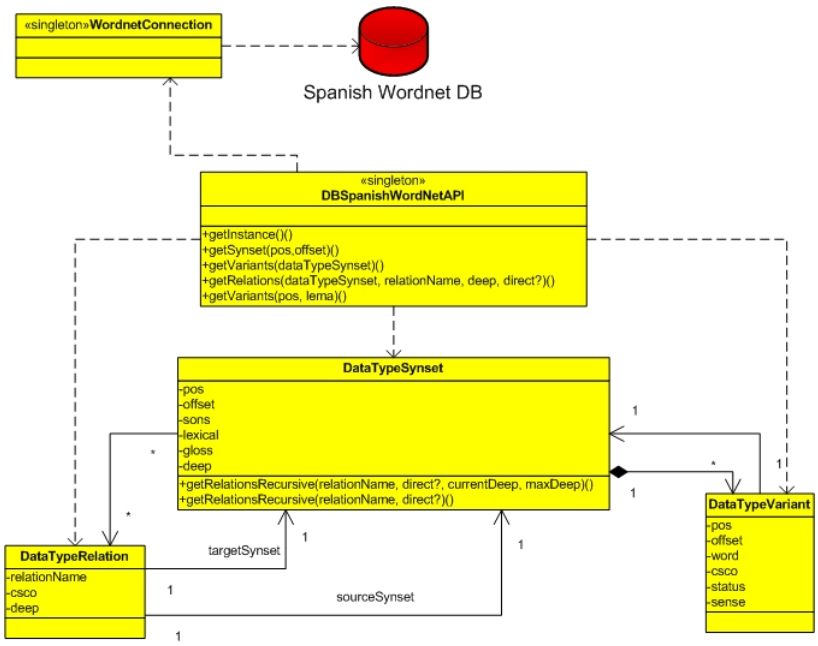


Figura B.4.: Diseño API WordNet

mento completo. Es importante mencionar que *InfoDoc* extiende *uima.tcas.DocumentAnnotation* la cual brinda funcionalidad para etiquetar (anotar) la totalidad del texto en UIMA.

B.4. Diseño del componente API WordNet

Este componente abstrae y encapsula toda la funcionalidad de acceso a la base de datos de Spanish WordNet. Básicamente define estructuras de datos que representa el esquema de la base de datos y un conjunto de métodos para explorarla. Su realización en un módulo independiente facilita su reutilización. Otra ventaja es que al tener un esquema muy semejante a los demás WordNet's se puede adaptar fácilmente a estos con muy pocas modificaciones.

La clase *DataTypeSynset* representa un concepto de la realidad. Los objetos de la clase *DataTypeVariant* asociados a un objeto de tipo *DataTypeSynset* representan los sinónimos del concepto. *DataTypeRelation* posee dos referencias a *DataTypeSynset* (ver Figura B.4) y representa una relación semántica binaria entre estos dos conceptos.

B.5. Diseño del componente de Utilidades

Este componente brinda un conjunto de funcionalidades comunes a los distintos componentes que forman parte del sistema ISRT.

RunCommandFreeling encapsula funcionalidades de la herramienta Freeling. Los métodos públicos presentes en *RunCommandFreeling* son:

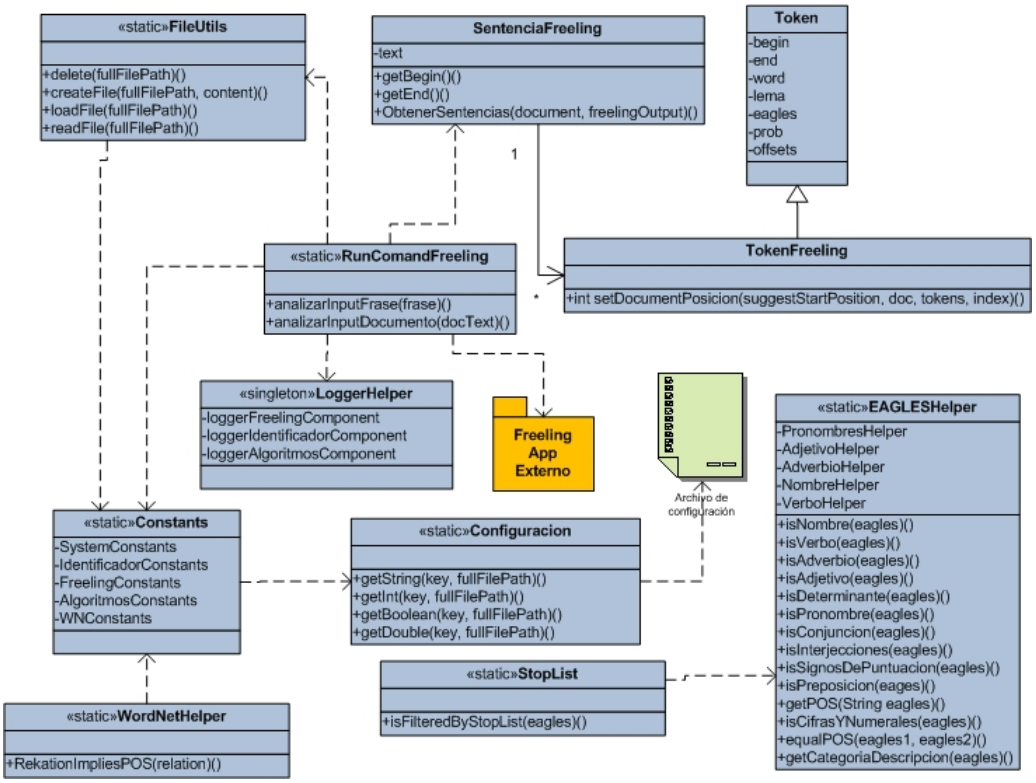


Figura B.5.: Diseño del componente de Utilidades

Algorithm B.5 Métodos públicos de RunCommandFreeling

public static Collection<SentenciaFreeling> analizarInputFrase(String frase)
public static Collection<SentenciaFreeling> analizarInputDocumento(String docText)

Algorithm B.6 Código de invocación de Freeling

```
//Comando para ejecutar
String comandoStr = "cmd.exe /C analyzer.exe -f es.cfg < FULL_PATH_FILE";
//Ejecución
Process proc = Runtime.getRuntime().exec(comandoStr);
```

La clase *Constants* brinda los valores constantes y parámetros del sistema, encapsula la complejidad de acceder a archivos físicos en el sistema operativo, y leer correctamente los valores.

EAGLESHelper contiene un conjunto de métodos de ayuda para trabajar con las etiquetas EAGLES.

StopList determina para una etiqueta EAGLES si esta se considerará para el análisis y búsqueda de relaciones semánticas.

SentenciaFreeling y *TokenFreeling* representan la salida del análisis morfológico de un texto generado por la aplicación externa Freeling.

C. Pruebas del sistema

En este anexo se completara la información sobre las pruebas realizadas al sistema, se presentaran las frases de consulta utilizadas, los documentos, y los resultados obtenidos para cada unos de los documentos.

C.1. Frases y Documentos

Como fue descripto en el capítulo 6, *Pruebas realizadas,* las pruebas fueron realizadas por grupos. Para esto se utilizaron 6 grupos, de los cuales 5 de ellos están formados por 3 documentos relacionados con 4 frases de consulta y 1 está formado por 3 documentos y las 20 frases de los conjuntos anteriores.

A continuación presentaremos la composición de los grupos, los documentos que lo conforman, las frases usadas en sus pruebas y los significados utilizados para la desambiguación manual.

- Grupo 1
  - Documentos
    - Calentamiento global y sus efectos en el clima
      - Oraciones 96
      - Oraciones con *offset* 89
      - Palabras 2124
      - Palabras con *offset* 859
      - Tokens 2322
      - *Offsets* 4084
    - Efecto invernadero petróleo
      - Oraciones 84
      - Oraciones con *offset* 84
      - Palabras 2056
      - Palabras con *offset* 880
      - Tokens 2262
      - *Offsets* 4921
    - El cenit del petróleo

- ◇ Oraciones 126
- ◇ Oraciones con *offset* 125
- ◇ Palabras 3731
- ◇ Palabras con *offset* 1511
- ◇ Tokens 4106
- ◇ *Offsets* 7737

- Frases
  - *Problemas del calentamiento global*
    - ◇ *problemas* [10340761,10, 11][1], Estado de dificultad que necesita resolverse, siempre tengo dificultades para localizarlo.
    - ◇ *calentamiento* [09740371,1,2], DPHH[2]
    - ◇ *global*[00493390, 3, 7], Estado de dificultad que necesita resolverse, siempre tengo dificultades para localizarlo.
  - *El cambio climático*
    - ◇ *cambio*[05441797, 7, 11], Acontecimiento que sucede cuando algo pasa de una estado o fase a otro: "el objetivo del cambio era aumentar las ventas"; "la tormenta provocó un cambio negativo"
    - ◇ *climático*[02805865, 1, 1], DPHH.
  - *Efecto invernadero*
    - ◇ *efecto*[07766144, 3, 3], Fenómeno que resulta y que está causado por otro fenómeno previo, las consecuencias de la tormenta fueron desastrosas.
    - ◇ *invernadero*[02770699,1,2], DPHH.
  - *Petróleo*
    - ◇ *petróleo*[10527530,1,2], Combustible derivado de los restos de organismos preservados en las rocas de la corteza terrestre con un alto contenido de carbono e hidrógeno.

- Grupo 2
  - Documentos
    - Crisis económica
      - ◇ Oraciones 107
      - ◇ Oraciones con *offset* 107

[1] *palabra [X,Y,Z]* - Donde X = *offset* seleccionado, Y = nro. *offset* y Z = cantidad de significados posibles.
[2] Determinado por Hiperónimos e Hipónimos

        - Palabras 2756
        - Palabras con *offset* 1210
        - Tokens 3086
        - *Offsets* 6587
    - Economía en Uruguay
        - Oraciones 50
        - Oraciones con *offset* 47
        - Palabras 1475
        - Palabras con *offset* 613
        - Tokens 1646
        - *Offsets* 3418
    - Elementos de macroeconomía y política económica
        - Oraciones 171
        - Oraciones con *offset* 171
        - Palabras 5551
        - Palabras con *offset* 2538
        - Tokens 6163
        - *Offsets* 13443
- Frases
    - *Aumento en los precios*
        - *Aumento* [00233535,1,9], Acción de incrementar algo.
        - *precios* [09576753,5,6], Cantidad de dinero que se necesita para comprar o vender algo, el precio del coche es de 24000 euros a precio de coste.
    - *Política económica*
        - *política*[04673837,3,5], Conjunto de doctrinas sobre el gobierno de los estados y los asuntos públicos en general, la política no me interesa.
        - *económica*[01800273,3,7], DPHH.
    - *Crisis financiera*
        - *crisis* [10040291,2,3], Definición Situación delicada o de gran dificultad: durante la crisis económica estuvieron en bancarrota.
        - *financiera*[], No existe en WordNet

        - *Oferta y Demanda*
            - *Oferta*[09726614,3,3], Acción de ofrecer bienes y servicios.
            - *demanda*[09725817,6,6], Cantidad de productos y servicios que solicita el mercado: los automóviles hicieron bajar la demanda de carros; siempre hay demanda de educación.

- Grupo 3
    - Documentos
        - Gripe
            - Oraciones 18
            - Oraciones con *offset* 18
            - Palabras 739
            - Palabras con *offset* 316
            - Tokens 826
            - *Offsets* 1392
        - Epidemias y pandemias
            - Oraciones 43
            - Oraciones con *offset* 43
            - Palabras 961
            - Palabras con *offset* 428
            - Tokens 1076
            - *Offsets* 1907
        - Gripe A
            - Oraciones 56
            - Oraciones con *offset* 56
            - Palabras 1577
            - Palabras con *offset* 667
            - Tokens 1735
            - *Offsets* 3233
    - Frases
        - *Gripe*
            - *Gripe* [10159752,1,1], DPHH.
        - *Enfermedades contagiosas*

        - *Enfermedades* [10129713,2,2] Alteración de la salud o funcionamiento anormal de algo.
        - *contagiosas* [01249364,2,3] DPHH.
    - *Epidemias y pandemias*
        - *Epidemias* [05531449,1,1] DPHH.
        - *pandemias* [05531611,1,1] DPHH.
    - *Muertes ocasionadas por enfermedades*
        - *Muertes* [05479076,2,7] Definición Fin de la vida: "su muerte fue una sorpresa horrible; cuando mueras tu capital pasará a tus nietos".
        - *ocasionadas* [01799148, 3,3] DPHH.
        - *enfermedades* [10129713,2,2] Alteración de la salud o funcionamiento anormal de algo.

- Grupo 4
    - Documentos
        - Futuro de la inteligencia artificial
            - Oraciones 32
            - Oraciones con *offset* 32
            - Palabras 1033
            - Palabras con *offset* 401
            - Tokens 1138
            - *Offsets* 2038
        - Que podemos esperar del futuro
            - Oraciones 50
            - Oraciones con *offset* 50
            - Palabras 771
            - Palabras con *offset* 339
            - Tokens 855
            - *Offsets* 1693
        - El futuro
            - Oraciones 39
            - Oraciones con *offset* 39
            - Palabras 891

        - Palabras con *offset* 397
        - Tokens 996
        - *Offsets* 2187
    - Frases
        - *Ciencia ficción*
            - *ciencia* [04596663,2,2] Dominio del conocimiento que se obtiene mediante el estudio sistemático de la naturaleza.
            - *ficción* [04800434,1,2] Composición literaria que narra sucesos imaginarios.
        - *El hombre del próximo milenio*
            - *hombre* [07392506,6,6] Uso genérico de la palabra para referirse a cualquier ser humano.
            - *próximo*[00417100,3,6] DPHH.
            - *milenio* [10866466,1,1] Espacio temporal de mil años.
        - *Las tecnologías y ciencias del futuro*
            - *tecnologías* [04660658,2,2] DPHH.
            - *ciencias* [04596663,2,2] Dominio del conocimiento que se obtiene mediante el estudio sistemático de la naturaleza.
            - *futuro* [10849807,2,2] Tiempo que aún tiene que ocurrir, el futuro de los universitarios es incierto, futuro negro.
        - *El tercer milenio*
            - *tercer* [02099122,1,1] DPHH.
            - *milenio* [10866466,1,1] Espacio temporal de mil años.

- Grupo 5
    - Documentos
        - Declaración de los derechos humanos
            - Oraciones 120
            - Oraciones con *offset* 88
            - Palabras 1897
            - Palabras con *offset* 819
            - Tokens 2155
            - *Offsets* 4129
        - Derechos humanos

        - Oraciones 136
        - Oraciones con *offset* 128
        - Palabras 3918
        - Palabras con *offset* 1739
        - Tokens 4505
        - *Offsets* 9325
    - Derechos inherentes a la defensa los derechos humanos
        - Oraciones 84
        - Oraciones con *offset* 75
        - Palabras 872
        - Palabras con *offset* 331
        - Tokens 996
        - *Offsets* 1675
- Frases
    - *Derechos humanos*
        - *Derechos* [04030305,1,3] Conjunto de normas que regulan las relaciones sociales, tienes derecho a un descanso de cinco minutos.
        - *humanos* [00383901,1,7] DPHH.
    - *La democracia, la igualdad y la justicia*
        - *democracia* [04711142,1,2] Doctrina política en la que el pueblo puede intervenir en el gobierno.
        - *igualdad* [03730389,3,5] Calidad o estado de ser igual en cantidad, medida, valor o estatus.
        - *justicia* [03801117,3,3] Cualidad de ser justo o imparcial.
    - *Obligaciones de los estados*
        - *Obligaciones* [00731811,1,4] Fuerza social que somete a las obligaciones y a los cursos de acción que esa fuerza requiere: “debemos inculcar a nuestros hijos el sentido del deber”.
        - *estados* [06074189,3,6] Grupo de gente organizada políticamente bajo un mismo gobierno, el estado ha elegido un nuevo presidente; el país votó al partido demócrata.
    - *Libertad*
        - *libertad* [10079933,1,4] Condición de ser libre; poder para actuar, hablar o pensar sin restricciones impuestas desde el exterior.

- Grupo GDNR
  - Documentos
    - Donde el diablo perdió la parabólica
      - Oraciones 35
      - Oraciones con *offset* 35
      - Palabras 1081
      - Palabras con *offset* 396
      - Tokens 1198
      - *Offsets* 2001
    - Más allá de la alharca
      - Oraciones 27
      - Oraciones con *offset* 27
      - Palabras 978
      - Palabras con *offset* 406
      - Tokens 1103
      - *Offsets* 1853
    - Terra incógnita
      - Oraciones 65
      - Oraciones con *offset* 65
      - Palabras 1359
      - Palabras con *offset* 583
      - Tokens 1567
      - *Offsets* 2828
  - Frases
    - Se usan las frases de los conjuntos anteriores con los mismos significados.

C.2. Resultados

C.2.1. Resultados Grupo 1

		Algoritmo de Selección 1			Algoritmo de Selección 2			Algoritmo de Selección 3		
Documento	Frase	IC	II	NI	IC	II	NI	IC	II	NI
1	1	8	2	3	6	1	5	10	5	1
	2	6	4	4	3	1	7	3	5	7
	3	2	2	0	1	1	1	1	1	1
	4	1	0	0	1	0	0	1	0	0
2	1	2	8	0	0	0	2	2	6	0
	2	5	5	2	5	9	2	6	13	1
	3	10	0	2	10	0	2	11	0	1
	4	5	0	0	5	0	0	4	0	1
3	1	8	2	12	10	0	10	14	16	6
	2	10	0	22	30	21	2	23	32	9
	3	6	4	1	6	8	1	6	4	1
	4	10	0	3	13	1	0	11	0	2
	Totales	73	27	49	90	42	32	92	82	30

Figura C.1.: Resultados Grupo 1

C.2.2. Resultados Grupo 2

		Algoritmo de Selección 1			Algoritmo de Selección 2			Algoritmo de Selección 3		
Documento	Frase	IC	II	NI	IC	II	NI	IC	II	NI
1	1	5	5	0	4	1	1	5	8	0
	2	9	1	4	12	4	1	12	4	1
	3	10	0	3	13	0	0	13	2	0
	4	7	3	3	6	0	4	8	0	2
2	1	7	3	1	4	0	4	8	4	0
	2	7	3	0	6	0	1	6	0	1
	3	1	2	0	1	0	0	1	1	0
	4	5	0	0	2	0	3	2	0	3
3	1	7	3	1	7	11	1	8	34	0
	2	6	4	15	21	9	0	17	6	4
	3	1	9	0	1	1	0	1	3	0
	4	10	0	17	27	9	0	27	10	0
	Totales	75	33	44	104	35	15	108	72	11

Figura C.2.: Resultados Grupo 2

C.2.3. Resultados Grupo 3

		Algoritmo de Selección 1			Algoritmo de Selección 2			Algoritmo de Selección 3		
Documento	Frase	IC	II	NI	IC	II	NI	IC	II	NI
1	1	9	0	2	9	0	2	7	0	4
	2	3	7	3	3	0	3	2	0	4
	3	5	0	0	5	0	0	4	0	1
	4	1	9	0	0	3	1	0	3	1
2	1	10	0	0	10	3	0	9	0	1
	2	8	2	1	3	0	6	6	0	3
	3	10	0	10	18	1	2	18	1	2
	4	2	8	1	0	1	3	1	13	2
3	1	10	0	2	11	0	1	10	0	2
	2	4	6	0	0	0	4	0	1	4
	3	10	0	9	16	0	3	16	0	3
	4	0	10	0	0	3	0	0	10	0
	Totales	72	42	28	75	11	25	73	28	27

Figura C.3.: Resultados Grupo 3

C.2.4. Resultados Grupo 4

		Algoritmo de Selección 1			Algoritmo de Selección 2			Algoritmo de Selección 3		
Documento	Frase	IC	II	NI	IC	II	NI	IC	II	NI
1	1	1	1	1	1	0	1	1	0	1
	2	1	9	0	0	0	1	1	6	0
	3	3	5	0	3	0	0	3	3	0
	4	0	3	0	0	0	0	0	0	0
2	1	0	4	0	0	3	0	0	3	0
	2	1	9	0	0	1	1	1	8	0
	3	0	10	0	0	0	0	0	7	0
	4	3	5	0	3	4	0	3	5	0
3	1	0	5	1	0	4	1	0	4	1
	2	2	3	0	1	0	1	2	2	0
	3	6	4	0	4	0	2	6	3	0
	4	3	1	0	1	0	2	1	0	2
	Totales	20	59	2	13	12	9	18	41	4

Figura C.4.: Resultados Grupo 4

C.2.5. Resultados Grupo 5

		Algoritmo de Selección 1			Algoritmo de Selección 2			Algoritmo de Selección 3		
Documento	Frase	IC	II	NI	IC	II	NI	IC	II	NI
1	1	10	0	33	43	1	0	41	1	2
	2	3	2	0	2	0	1	2	0	1
	3	4	6	0	4	11	0	4	28	0
	4	10	0	8	14	1	4	13	0	5
2	1	10	0	66	64	19	12	57	15	19
	2	4	6	0	1	0	3	4	4	0
	3	5	5	6	10	13	1	10	53	1
	4	5	5	2	5	18	2	7	0	0
3	1	9	1	43	15	7	37	18	7	34
	2	0	5	0	0	0	0	0	0	0
	3	4	5	0	4	0	0	4	7	0
	4	6	2	0	6	0	0	6	0	0
	Totales	70	37	158	168	70	60	166	115	62

Figura C.5.: Resultados Grupo 5

C.2.6. Resultados Grupo GDNR

		Algoritmo de Selección 1			Algoritmo de Selección 2			Algoritmo de Selección 3		
Documento	Frase	IC	II	NI	IC	II	NI	IC	II	NI
1	G1-1	0	1	0	0	0	0	0	0	0
	G1-2	0	9	0	0	1	0	0	2	0
	G1-3	0	1	0	0	0	0	0	0	0
	G1-4	0	0	0	0	0	0	0	0	0
	G2-1	0	5	0	0	0	0	0	1	0
	G2-2	0	2	0	0	0	0	0	0	0
	G2-3	0	0	0	0	0	0	0	0	0
	G2-4	0	3	0	0	0	0	0	1	0
	G3-1	0	0	0	0	0	0	0	0	0
	G3-2	0	1	0	0	0	0	0	0	0
	G3-3	0	0	0	0	0	0	0	0	0
	G3-4	0	0	0	0	0	0	0	0	0
	G4-1	0	0	0	0	0	0	0	0	0
	G4-2	0	1	0	0	0	0	0	0	0
	G4-3	2	4	0	0	0	2	2	0	0
	G4-4	0	1	0	0	0	0	0	0	0
	G5-1	2	0	0	2	0	0	2	0	0
	G5-2	0	6	0	0	0	0	0	0	0
	G5-3	0	10	0	0	1	0	0	3	0
	G5-4	0	0	0	0	0	0	0	0	0
2	G1-1	0	1	0	0	0	0	0	1	0
	G1-2	0	6	0	0	1	0	0	2	0
	G1-3	0	6	0	0	1	0	0	2	0
	G1-4	0	0	0	0	0	0	0	0	0
	G2-1	0	4	0	0	0	0	0	2	0
	G2-2	0	3	0	0	0	0	0	0	0
	G2-3	1	1	0	0	0	1	0	0	1
	G2-4	0	3	0	0	0	0	0	1	0
	G3-1	0	0	0	0	0	0	0	0	0
	G3-2	0	2	0	0	0	0	0	0	0
	G3-3	0	1	0	0	0	0	0	0	0
	G3-4	0	9	0	0	0	0	0	0	0
	G4-1	0	0	0	0	0	0	0	0	0
	G4-2	0	0	0	0	0	0	0	0	0
	G4-3	0	1	0	0	0	0	0	0	0
	G4-4	0	0	0	0	0	0	0	0	0
	G5-1	0	1	0	0	0	0	0	0	0
	G5-2	0	8	6	0	0	6	6	0	0
	G5-3	0	10	0	0	2	0	0	4	0
	G5-4	1	0	0	1	0	0	0	0	1
3	G1-1	0	1	0	0	0	0	0	1	0
	G1-2	0	5	0	0	0	0	0	0	0
	G1-3	0	1	0	0	0	0	0	1	0
	G1-4	0	4	0	0	0	0	0	0	0
	G2-1	0	2	0	0	0	0	0	1	0
	G2-2	0	0	0	0	0	0	0	0	0
	G2-3	0	1	0	0	0	0	0	0	0
	G2-4	0	0	0	0	0	0	0	0	0
	G3-1	0	0	0	0	0	0	0	0	0
	G3-2	0	0	0	0	0	0	0	0	0
	G3-3	0	1	0	0	0	0	0	0	0
	G3-4	0	6	0	0	0	0	0	2	0
	G4-1	0	4	0	0	0	0	0	1	0
	G4-2	0	5	0	0	0	0	0	3	0
	G4-3	0	5	0	0	0	0	0	1	0
	G4-4	0	3	0	0	0	0	0	0	0
	G5-1	0	0	0	0	0	0	0	0	0
	G5-2	0	2	0	0	0	0	0	0	0
	G5-3	0	10	0	0	2	0	0	7	0
	G5-4	0	0	0	0	0	0	0	0	0
	Totales	6	150	6	3	8	9	10	36	2

Figura C.6.: Resultados Grupo GDNR

D. Otras herramientas

En este anexo presentaremos las herramientas que quedaron pendientes en el capítulo 5, *Desarrollo del Proyecto,* y que fueron de suma importancia en el desarrollo del trabajo.

LyX vs MikTex

Para la documentación del proyecto es necesario la utilización de un procesador de texto, la primera opción es la utilización de los procesadores de texto tradicionales como son Microsoft Office Word ó Open Office Writer, ambos serían una excelente elección ya que son herramientas poderosas en sus funcionalidades y ampliamente populares. Sin embargo, estas herramientas tienen en común que requieren gran esfuerzo para darle un estilo homogéneo a un documento de gran tamaño, es por esto que se buscó alguna herramienta que nos permitiera concentrarnos en el contenido del documento y no en el estilo.

LyX es un procesador de documentos que apoya el enfoque de escribir basándose en la estructura de los documentos WYSIWYM[1] y no simplemente el enfoque basado en la apariencia WYSIWYG[2] como los procesadores de texto tradicionales.

LyX combina el poder y flexibilidad de Tex/LaTex con la facilidad de uso de una completa interfaz gráfica. Esto resulta en un soporte de clase mundial para la creación de contenidos matemático (gracias un editor de ecuaciones completamente integrados) y la redacción de documentos estructurados como artículos académicos, tesis y libros.

MikTex es otro procesador de documentos con el mismo enfoque WYSIWYM, cuenta con una interfaz gráfica menos desarrollada que la de LyX pero con las mismas funcionalidades.

La elección de un procesador basado en el enfoque WYSIWYN se basó en el hecho que generan documentos de gran calidad y permiten centrarnos principalmente en el contenido, la desventaja de ambos es que tiene una curva de aprendizaje mayor a los procesadores tradicionales. De los dos procesadores probados se eligió la utilización de LyX sobre MikTex, las razones que apoyan la elección son las siguientes: La interfaz de LyX es más amigable y fácil de usar, brinda las mismas funciones y tiene una menor curva de aprendizaje. La ventaja de múltiples archivos de MikTex es absorbida por la utilización del control de cambios que provee LyX y las funcionalidades de edición de conflictos brindadas por SVN Tortoise.

[1] *What you see is what you mean*, lo que ves es lo que quieres.
[2] *What you see is what you get*, lo que ves es lo que obtienes.

Google Code Project Hosting + TortoiseSVN

Google Code Project Hosting, es un sitio web gratuito patrocinado y desarrollado por Google que brinda espacio para la creación de repositorios de proyectos. Google ofrece 100Mbytes (o más si es requerido) de espacio y usa Subversion[3] para el control de versiones.

TortoiseSVN, es un cliente Windows para Subversion, es fácil de usarlo y ofrece control de revisiones, de versiones y de código y un editor de conflictos. TortoiseSVN provee una interfaz simple y sencilla acoplada al explorador de Windows. Ha sido desarrollada con una licencia GPL, lo que significa que es totalmente gratuita y de código abierto.

La utilización de un sistema de versionado, no sólo para el código sino también para los documentos generados en el proyecto fue desde un principio incuestionable. La primera opción que estudiamos fue la utilización del Repositorio CVS que ofrece el InCo, pero tuvimos problemas de comunicación con los encargados que dificultaron la utilización de este *hosting*. En consecuencia empezamos a buscar un sustituto, de esta forma encontramos la combinación de Google Code Project Hosting + TortoiseSVN. La utilización de ambas herramientas en conjunto nos proveía todas las funcionalidades que el CVS tradicional más otras nuevas, además tendríamos control total tanto del *hosting* como del cliente. El resultado de la utilización de estas herramientas ha sido ampliamente satisfactorio.

Java

La aplicación fue implementada utilizando el lenguaje de programación Java, la principales características de Java son:

- **Simple**. Elimina la complejidad de los lenguajes como "C". Lenguaje orientado a objetos.
- **Familiar**. Como la mayoría de los programadores están acostumbrados a programar en C o en C++, la sintaxis de Java es muy similar al de estos.
- **Robusto**. Java administra la memoria de la computadora por ti. No te tienes que preocupar por apuntadores, memoria que no se esté utilizando, etc.
- **Seguro**. El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.
- **Portable**. Como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el interprete de Java.

[3]Subversion sistema para el control de versiones de código abierto, similar al CVS.

- **Independiente a la arquitectura**. Al compilar un programa en Java, el código resultante un tipo de código binario conocido como byte code. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.

- **Multithreaded**. Un lenguaje que soporta múltiples threads es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo.

- **Interpretado**. Java corre en máquina virtual, por lo tanto es interpretado.

- **Dinámico**. Java no requiere que compiles todas las clases de un programa para que este funcione. Si realizas una modificación a una clase Java se encarga de realizar un Dynamic Bynding o un Dynamic Loading para encontrar las clases.

Las razones que influyeron en la selección de Java como lenguaje de programación para el proyecto son las siguientes:

- Las características propias del lenguaje.

- Conocimiento del lenguaje.

- La plataforma Lavinia fue desarrollado en Java y un requerimiento es poder integrarnos a ésta.

- Freeling tiene una compilación para este lenguaje y un ejecutable que sirve de interfaz.

- Los scripts de WordNet están escritos para MySQL y aunque eso no impide la utilización de otros lenguajes el hecho que MySQL fuera creado por Sun Microsystem creadora de Java apoya la elección.

- UIMA brinda un framework para el desarrollo de sistemas de análisis léxicos que funciona en Java.

MySQL

MySQL es una herramienta de gestión de base de datos relacional, desarrollado por Sun Microsystem (mismo desarrollador de Java), se ofrece como un software libre bajo una licencia dual, GNU GPL o una específica para usos privados.

El uso de MySQL fue determinado por el hecho que los scripts de base de datos de WordNet estaban generados para este administrador de base de datos, sumado al hecho que es un poderoso y reconocido administrador y ampliamente compatible con Java. Por lo antes mencionado no tuvimos razones para migrar los scripts. Por lo que el desarrollo del proyecto se apoyó en este manejador de base de datos.

Eclipse

Eclipse es un entorno de desarrollo integrado[4] de código abierto multiplataforma para el desarrollo de proyectos. Brinda diversos servicios que facilitan el desarrollo de diversos tipos de proyectos.

Fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge, actualmente es desarrollado por la Fundación Eclipse, una organización independiente sin fines de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Para el desarrollo del proyecto utilizamos la última versión disponible de Eclipse, Eclipse Ganymedes, el cual brinda múltiples elementos que facilitan el desarrollo del proyecto y acepta una inmensa variedad de plugins que pueden ser agregados al IDE para aumentar sus funcionalidades. Dentro de estos plugins se encuentra un Framework para UIMA que se acopla al Eclipse y brinda numerosas funcionalidades para el desarrollo de sistemas UIMA. Este último punto fue de gran peso a la hora de seleccionar Eclipse como el IDE de desarrollo.

Apache Tomcat

Uno de los requerimientos del proyecto fue poder integrar nuestra solución al framework Lavinia. Lavinia es un aplicación web y por lo tal requiere de un servidor web para su funcionamiento.

Apache Tomcat[6] es un servidor web con soporte de servlets y JSPs desarrollado bajo el proyecto Jakarta en la *"Apache Software Foundation"*. Es un proyecto *open source*, implementa la especificación de las tecnologías Java Server y Java Server Pages brindada por *"Java Comunity Process"*.

La elección de Apache Tomcat como servidor web en el desarrollo del proyecto se debió a que Lavinia fue desarrollada con el mismo, lo que minimiza el riesgo de incompatibilidades. A esto se suma el hecho que actualmente UIMA está bajo la administración de Apache.

[4]IDE (sigla en inglés).

E. Integración a Lavinia

El propósito de este anexo es explicar cómo puede ser integrado el sistema desarrollado a Lavinia y como una vez integrado éste puede ser utilizado.

E.1. Incorporación a Lavinia

Si bien Lavinia provee una interfaz para el agregado de componentes, está no puede ser utilizada para la integración de nuestro sistema. Esto se debe a que nuestro sistema hace uso de herramientas UIMA no presentes en la interfaz. A continuación describiremos los pasos necesarios para la integración del sistema a Lavinia.

1. Configuración de los recursos[1]

   Modificar el archivo:

   <apacheDir>\webapps\Lavinia\WEB-INF\resources\LaviniaPaths.xml

   y agregar las siguientes entradas:

   <FreelingExe2punto0 description="Freeling2.0" externo="true">

   C:/FreeLing/analyzer.exe

   </FreelingExe2punto0>

   <FreelingExe2punto0ConfEsp description="Freeling2.0" externo="true">

   C:/FreeLing/es.cfg

   </FreelingExe2punto0ConfEsp>

   <ISRConfig description="Freeling2.0" externo="true">

   C:/Idetem/configuracion/application.properties

   </ISRConfig>

   Estas entradas permiten definir las rutas a los distintos recursos externos que utiliza el sistema. Un ejemplo del archivo *application.properties* se puede encontrar en la carpeta *<CDroot>\Jars\JarsComponentes\Configuración\IdTem.*

[1] Verificar que el archivo de configuración no esté configurado para usar la ventana de desambiguación.

2. Instalación de los JAR's de los componentes y recursos

   Para poder ejecutar el sistema desde Lavinia[7] se tendrán que copiar los JAR's contenidos en las carpetas

   <CDroot>\Jars\JarsComponentes

   <CDroot>\Jars\JarsRecursos

   al directorio web de Lavinia (*<apacheDir>\webapps\Lavinia\WEB-INF\lib*)

3. Instalación de los descriptores

   Luego de la instalación de los componentes se deberán copiar las carpetas contenidas en

   <CDroot>\LaviniaDescriptors

   al directorio

   <apacheDir>\webapps\Lavinia\WEB-INF\resources\engines.

4. Agregar los componentes

   En el archivo *LaviniaComponents.xml* que se encuentra en

   <apacheDir>\webapps\Lavinia\WEB-INF\resources

   se deben agregar tres entradas del nodo *"components"* como se muestra a continuación:

   <components>

   .

   <component simple="true">Freeling2Annotation</component>

   <component simple="true">IdentificadorAnnotation</component>

   <component simple="true">AlgoritmosAnnotation</component>

   .

   </componets>

5. Levantar el servidor

6. Crear el flujo ISRT

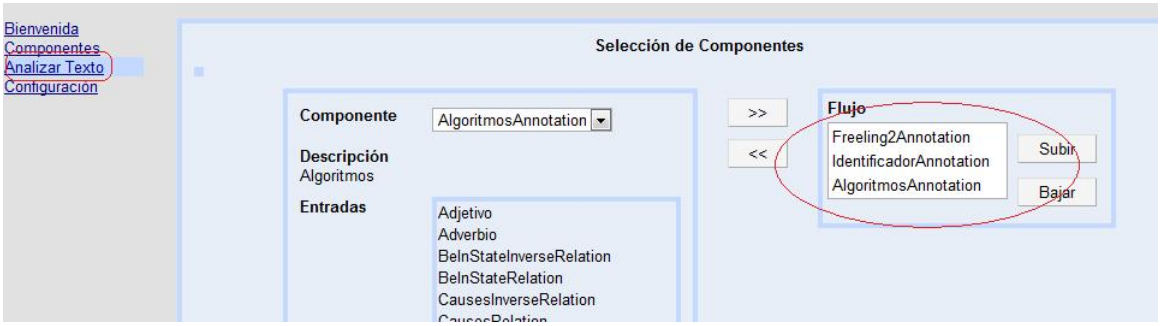
   Ingresar a Lavinia y crear el Flujo ISRT.

a) El nombre: ISRT

b) Los autores: PLN-InCo

c) Descripción: Identificador de Segmentos Relacionados Temáticamente

d) Agregar los componentes: Freeling2Annotator, IdentificadorAnnotator y AlgoritmoAnnotator en este orden.



Si se siguieron correctamente los pasos antes descriptos, el sistema quedó correctamente integrado a Lavinia y pronto para su utilización.

E.2. Utilización en Lavinia

A continuación se presentará un ejemplo de utilización del sistema desde Lavinia.

1. Una vez ingresado a Lavinia se deberá acceder a la pantalla *"Analizar Texto"*

2. En la pantalla de *"Selección de Componentes"* se deberá seleccionar el flujo ISRT (creado durante la instalación) y presionar *siguiente*.

3. En la pantalla *"Configuración de Parámetros"*, el usuario podrá seleccionar diferentes valores para los parámetros del componente *IdentificadorAnnotator* especialmente el usuario deberá ingresar el valor de la frase de consulta que será utilizada para el análisis del texto, ver Figura E.1.

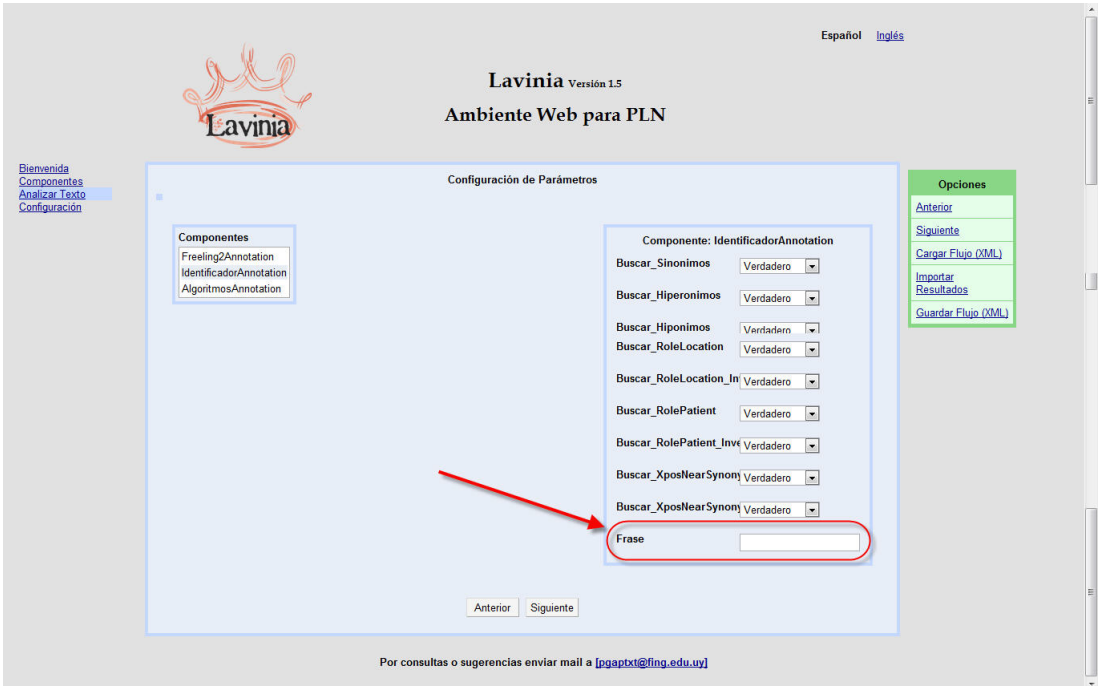


Figura E.1.: Pantalla de configuración de Lavinia

4. Luego de configurado los parámetros e ingresado la frase de consulta. Se puede acceder a la pantalla de *"Análisis de Texto"*, en esta pantalla el usuario podrá ingresar el texto a analizar, seleccionar los valores de retorno que le interesa y finalmente analizar el texto. En la Figura E.2, se puede ver un ejemplo de ejecución.

Por más información de cómo crear flujos, seleccionar las etiquetas a marcar y otras configuraciones, consultar el manual de Lavinia[7].

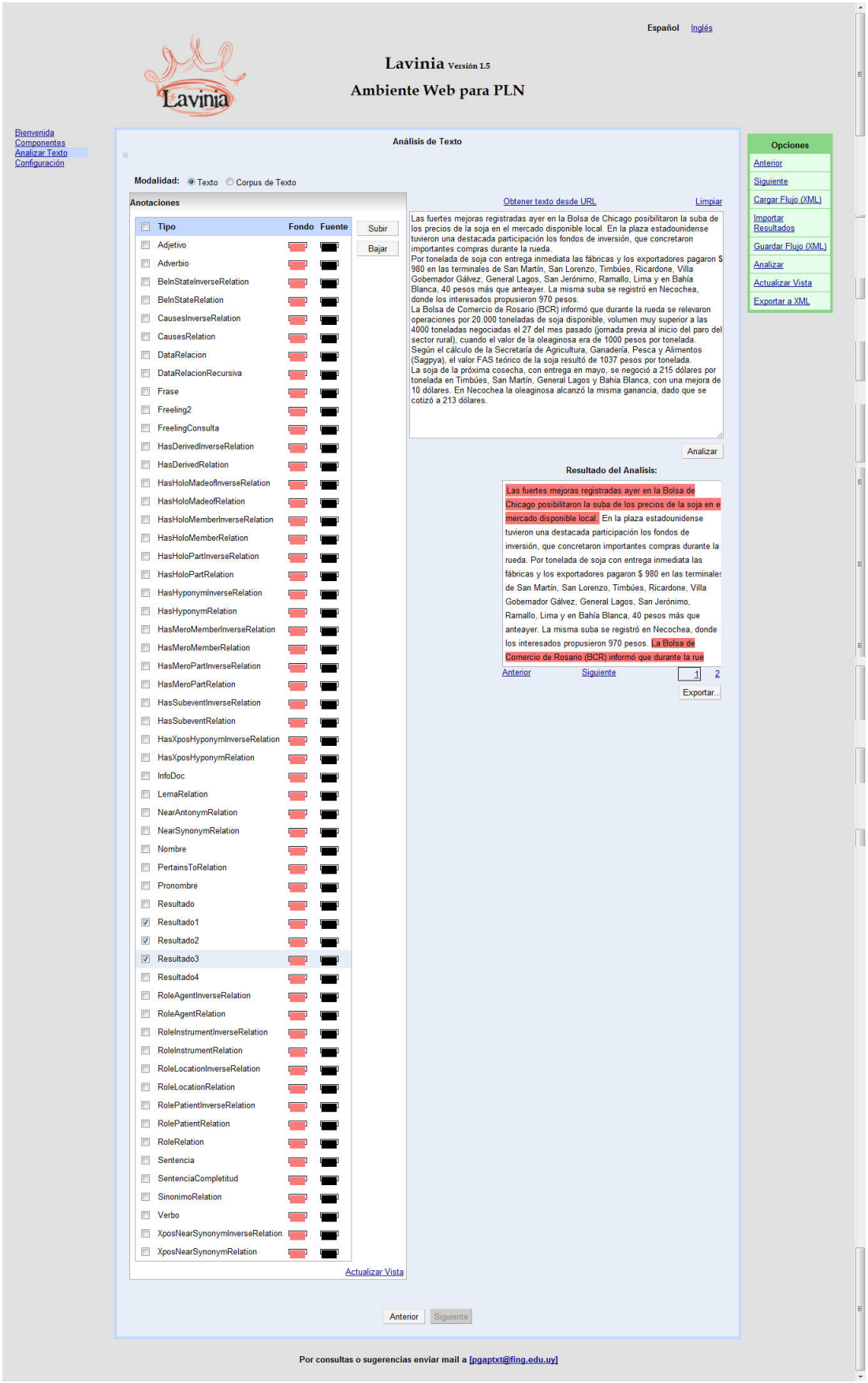


Figura E.2.: Ejecución del sistema ISRT desde Lavinia

F. Instalación del entorno de desarrollo

Este anexo tiene como objetivo explicar los pasos a seguir para configurar el ambiente de desarrollo.

1. El primer paso es descargar e instalar las herramientas necesarias:

   a) Java JDK (*Java Development Kit*) 1.6.

   b) IDE (*Integrated Development Environment*) Eclipse Ganymed.

   c) Manejador de base de datos MySql 5.1.32.

   d) Paquetes de UIMA.

   e) Freeling 2.0.

   f) Servidor web Apache-Tomcat 6.0.

   g) Lavinia.

   Las herramientas Apache-Tomcat y Lavinia solamente serán necesarias si se desea ejecutar Lavinia en la maquina local. Otra consideración es que se podrá utilizar otras versiones de las herramientas, las versiones especificadas corresponden con las utilizadas en el desarrollo del proyecto.

2. Luego se debe importar el *workspace* que se encuentra en la carpeta Fuentes (ver G, *Contenido del CD de entrega*) y configurar el proyecto. Se tendrá que agregar la variable (*classpath*) UIMA_HOME para que referencie a la carpeta que contiene los paquetes de UIMA. Los paquetes se pueden encontrar en la ruta *<CDroot>\Recursos\UIMA\apache-uima* dentro del CD de entrega. También se tendrá que definir la variable (*classpath*) IdTemResource_Home para que referencie a la carpeta que se encuentra en *<CDroot>\Sources\workspace\resources*.

3. En caso de querer ejecutar o debuguear el sistema se debe crear la base de datos ejecutando el script *SpanishWordNetBackup.sql* contenido en la carpeta *<CDroot>\Sources\workspace\resources*.

G. Contenido del CD de entrega

En el CD de entrega se encuentra toda la documentación, código fuente, scripts de la base de datos Spanish WordNet y otros recursos relacionados con el proyecto. Es objetivo de este anexo es explicar el contenido y estructura de las carpetas del CD.

En la raíz del CD se encuentra este documento (TesisISRT - Anexos.pdf), el informe principal (TesisISRT - Informe Principal.pdf) y varias carpetas que se detallaran a continuación.

- Fuentes: Contiene todo el código fuente de la aplicación. El mismo está estructurado de forma que pueda ser importado desde el Eclipse.
- Jars: Contiene diversos JAR's que permiten ejecutar la aplicación. Las subcarpetas son:
  - JarsComponentes: Contiene los archivos JAR's de los distintos componentes
    - Freeling2Annotation. jar
    - IdTemUtils.jar
    - WordNetAPI.jar
    - IdentificadorAnnotation.jar
    - AlgoritmosAnnotation.jar
  - JarsRecursos: Contiene archivos JAR's de recursos utilizados por la aplicación.
- LaviniaDescriptors: Contiene los archivos de configuración necesarios para la integración de los componentes en Lavinia.
- Pruebas: Contiene todos los recursos referentes a las pruebas realizadas en el sistema. Las carpetas contenidas son nemotécnicas, por lo cual se mencionará cada una.
  - Documentos Originales: Documentos usados en las pruebas sin ningún procesamiento.
  - Documentos Marcados: Contiene para los distintos grupos, los documentos marcados por un humano.
  - PRUEBAS - desambiguación manual: Contiene para los distintos grupos los documentos marcados por el sistema, desambiguando la frase de consulta de forma manual (ventana de desambiguación).

    - PRUEBAS - desambiguación significado más frecuente: Contiene para los distintos grupos los documentos marcados por el sistema, utilizando el significado más frecuente como método de desambiguación.

- Recursos: Esta carpeta contiene distintos recursos que necesita la aplicación para ejecutarse.
    - SpanishWordNet: Contiene el script para crear la base de datos Spanish WordNet.
    - Freeling: Contiene el programa Freeling2.
    - UIMA: Contiene los JAR's para poder utilizar UIMA.

Bibliografía

[1] Analizador lingüístico Freeling
http://garraf.epsevg.upc.es/freeling
Último acceso: 14/11/2009

[2] WordNet
http://wordnet.princeton.edu
Último acceso: 14/11/2009

[3] EuroWordNet
http://www.illc.uva.nl/EuroWordNet
Último acceso: 14/11/2009

[4] Spanish WordNet
Último acceso: 14/11/2009

[5] Apache UIMA
http://incubator.apache.org/uima
Último acceso: 14/11/2009

[6] Apache Tomcat
http://tomcat.apache.org
Último acceso: 14/11/2009

[7] Lavinia - Grupo de Procesamiento de Lenguaje Natural - UdelaR
http://www.fing.edu.uy/inco/grupos/pln/lavinia.html
Último acceso: 14/11/2009

[8] Java Swing
http://java.sun.com/javase/6/docs/technotes/guides/swing
Último acceso: 14/11/2009

[9] TREC (*Text REtrival Conference*)
http://trec.nist.gov
Último acceso: 14/11/2009

[10] NIST (*National Institute of Standards and Technology)*
http://www.nist.gov
Último acceso: 14/11/2009

[11] MUC (*Message Understanding Conference*)
http://www-nlpir.nist.gov/related_projects/muc
Último acceso: 14/11/2009

[12] Enciclopédia Encarta
http://es.encarta.msn.com
Último acceso: 14/11/2009

[13] Wikipedia
http://www.wikipedia.org
Último acceso: 14/11/2009

[14] WordReference
http://www.wordreference.com
Último acceso: 14/11/2009

[15] Wiki Lengua del español
http://www.wikilengua.org
Último acceso: 14/11/2009

[16] Real Academia Española.
http://www.rae.es/
Último acceso: 14/11/2009

[17] Grupo de Procesamiento de Lenguaje Natural - UdelaR
http://www.fing.edu.uy/inco/grupos/pln
Último acceso: 14/11/2009

[18] Corin
http://www.fing.edu.uy/inco/grupos/pln/publicaciones/Uruguay-Grassi-Wonsever-et-al_2_.pdf
Último acceso: 14/11/2009

[19] Proyecto RicoTerm 2
http://ricoterm.iula.upf.edu/2/
Último acceso: 14/11/2009

[20] Design Patterns. Elements of Reusable Object Oriented Software - *Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (2007)*

[21] Speech and Language Processing - *Daniel Jurafsky, James H. Martin - Prentice-Hall (2000)*

[22] Automatic Topic Identification Using Ontology Hierarchy - *Sabrina Tiun, Rosni Abdullah, Tang Enya Kong (2001)*
http://utmk.cs.usm.my/pdf/CICLing01.pdf
Último acceso: 14/11/2009

[23] Knowledge-based Automatic Topic Identification - *Chin-Yew Lin, Departamento de ingeniería Eléctrica, Universidad del Sur de California, Los Angeles, Estados Unidos (1995)*
http://acl.ldc.upenn.edu/P/P95/P95-1046.pdf
Último acceso: 14/11/2009

[24] The Effect of Using Hierarchical Classifiers in Text Categorization - *Stephen D'Alessio, Keitha Murray, Robert Schiaffino (2000)*
http://www.iona.edu/academic/artsscience/departments/computerscience/faculty/FacultyPublications/riao2000New.doc
Último acceso: 14/11/2009

[25] Using Frequently Occurring Words to Identify the Subject of a Document - *Offer Drori, Universidad hebrea de Jerusalén (2001)*
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.3339&rep=rep1&type=pdf
Último acceso: 14/11/2009

[26] Using WordNet for Word Sense Disambiguation to Support Concept Map Construction - *Institute for Human and Machine Cognition - Universidad EAFIT (2003)*
http://cmap.ihmc.us/Publications/ResearchPapers/SPIRE-2003-UsingWordnetforWordSenseDisambiguationtoSupportConceptMapConstruction.pdf
Último acceso: 14/11/2009

[27] Information Extraction: Techniques and Challenges - *Ralph Grishman, Departamento de Ciencias de la Computación, Universidad de Nueva York, Estados Unidos (1997)*
http://web.njit.edu/~ql23/teaching/cis634FTF/notes/Information_extraction_paper.pdf
Último acceso: 14/11/2009

[28] Identifying Topics by Position - *Chin-Yen Lin, Eduard Hovy. Departamento de Ciencias de la Computación, Universidad del Sur de California, Los Angeles, Estados Unidos (1997)*
http://www.mariapinto.es/ciberabstracts/Articulos/74.pdf
Último acceso: 14/11/2009

[29] Algoritmo de resolución de la anáfora pronominal en diálogos - *Patricio Martínez Barco, Grupo de Investigación en Procesamiento del Lenguaje y Sistemas de Información, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante (1999)*
http://www.sepln.org/revistaSEPLN/revista/24/24-articulo8.pdf
Último acceso: 14/11/2009

[30] Resolución de anáfora pronominal para el idioma español usando el método de conocimiento limitado - *Grigori Sidorov, Omar Olivas Zazueta Laboratorio de Lenguaje Natural y Procesamiento de Texto, Centro de Investigación en Computación, Instituto Politécnico Nacional (2006)*
http://ccc.inaoep.mx/~tec_lenguaje06/articulos/TLH06-paper11.pdf
Último acceso: 14/11/2009

[31] Ontología por Tom Gruber - *Encyclopedia of Database Systems (2007)*
http://tomgruber.org/writing/ontology-definition-2007.htm

[32] Definición de ontología como especificación del conocimiento - *Chantal Pérez (2002)*
http://elies.rediris.es/elies18/531.html
Último acceso: 14/11/2009

[33] Evaluación de Buscadores Web - *Marta Torres Magán, Universidad Carlos III de Madrid*
http://usuarios.lycos.es/webbuscadores/evaluacion_buscadores.pdf
Último acceso: 14/11/2009

[34] Definicion de EI brindada por Javier Aragón Zabalegui - Universidad Carlos III de Madrid
http://extraccioninformacion.iespana.es
Último acceso: 14/11/2009

[35] Multilingual Ontology-Based Lexicon for News Filtering - The TREVI Project - *Weigand, H. (1997)*

[36] Integrating Medical Terminologies with ONIONS Methodology - *Steve, G, A. Gangemi, D. Pisanelli (1998)*

[37] Formal Ontologies and Information Systems - *Guarino, N. (1998)*

[38] Learning to Extract Text-Based Information from the World Wide Web - *Stephen Soderland (1997)*