Universidad de las Ciencias Informáticas

Facultad 2

SUBSISTEMA DE ANÁLISIS SEMÁNTICO DE TEXTO PARA EL SISTEMA AUTOMATIZADO DE LA SUPERACIÓN PEDAGÓGICA EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Yanet Crespo Díaz
Eric Rodríguez Ochoa

Tutores: Dr.C. Febe Angel Ciudad Ricardo
Dr.C. Amed Abel Leiva Mederos
Ing. Walfrido Serrano Pérez

La Habana, junio de 2014
"Año 56 de la Revolución"

“O nosotros somos capaces de destruir con argumentos las ideas contrarias, o debemos dejar que se expresen. No es posible destruir ideas por la fuerza, porque esto bloquea cualquier desarrollo libre de la inteligencia."

Ernesto Guevara de la Serna

AGRADECIMIENTOS

De Yanet

Agradezco infinitamente a mi mamá que ha sido mi todo desde que vine al mundo, la cual me ha brindado todo el amor, cariño, confianza, fuerzas y ánimo que he necesitado para seguir adelante y llegar a realizar este sueño que desde la secundaria he tenido. Mami gracias por enseñarme a confiar más en mí, por alumbrarme en cada examen en los cuales compartías sentimentalmente conmigo, gracias por tus sabios consejos, gracias por esas llamadas telefónicas sin importar hora ni tiempo las cuales alegraban mi corazón, en fin, mil gracias por todo el sacrificio, preocupación y dedicación, te amo.

A mi papá por sus consejos, a mi hermano por ser el mejor hermano del mundo, a mi cuñada por dar a luz al otro amor de mi vida, mi sobrinito. A mi abuela que por problemas de salud no me acompaña personalmente pero yo sé que en este momento me desea lo mejor del mundo, te quiero mima. A toda mi familia por darme su apoyo incondicional, en especial a mi tío Ernesto y tía Belkis.

A Javy ya que en estos momentos obtiene el título de doble ingeniero, pues desde 1er año me ha acompañado en las buenas y en las malas, no importaba la asignatura. Mi vida te doy gracias por apoyarme, por ser tan comprensivo, por darme consuelo y fuerzas en los momentos más difíciles, por confiar en mí, por estar siempre ahí para mí. Te AMO.

A los viejones que vienen en representación de toda la familia de Granma, Dimi, Irialys, Viejona Mayor, Lisy y Lary, gracias a todos por su preocupación, apoyo y confianza, los quiero.

A Eric por ser tan buena persona y compartir conmigo este año de lucha para lograr alcanzar nuestro objetivo, sin dudas el mejor compañero de tesis. A Liset por su apoyo, ayuda y amistad incondicional.

A mi tutor Febe Angel por enseñaros, guiarnos y orientarnos por el mejor camino, a Amed por su apoyo y preocupación que estando en Villa Clara nos atendió incondicionalmente, a Walfrido que aunque no se encuentra en la Universidad también nos brindó su apoyo para que todo saliera bien.

A mis profesores de 5 años que batallaron cada día por hacernos mejores personas. A mis compañeros de aula, al team SASPED y amigos por su apoyo incondicional, por compartir juntos momentos de felicidad y tensión en estos 5 años, gracias por todo, nunca los olvidaré.

A Fidel y la Revolución por permitirnos graduarnos en esta Universidad de excelencia.

En general, a todos mil gracias, ya que sin ustedes este sueño no se hubiera podido realizar.

De Eric

Los agradecimientos son para mí lo más difícil en todo el documento, espero no se me quede nadie sin agradecer.

A mi familia, en especial a mi mamá que ya no está con vida junto a mí y a mi papá que es el mejor padre del mundo, los quiero. A mi hermano por sus consejos y sus buenas conversaciones. A mis tres abuelos por preocuparse por mí y haberme dado a los mejores padres. A mi novia Liset, por darme confianza y amor, por estar aquí a mi lado en todos estos años. A mi tía Arminda y a mi prima Adonay junto a su esposo por acogerme es su casa y hacerme sentir como si estuviese en mi casa. A la familia de Liset, Sofía, Hermes, Ismael, Marbelis y demás, por permitirme ser parte de ellos y acogerme como el novio de Liset.

A mi compañera de tesis y amiga Yanet, por confiar en mí como compañero de tesis y por su dedicación y abnegación todo este tiempo. A mis tutores Febe Angel, Amed y Walfrido por ser exigentes y guiarme por los caminos correctos en la investigación científica y por su confianza en mí. A Javier el novio de Yanet, por soportarme tanto tiempo, por poner carácter cuando estábamos algo preocupados con la investigación, por aconsejarnos y dar su visto bueno. A mis compañeros de cuarto y al grupo de desarrollo SASPED.

A las compañeras de apartamento de Liset en especial a Mirnerys, Claudia María y Claudia Celeste por su apoyo. A la Revolución y a Fidel por darme esta oportunidad. A todos los que de una forma u otra hicieron posible que yo llegara hasta aquí.

DEDICATORIA

De Yanet

Dedico este trabajo a toda mi familia y en especial a una persona que amo mucho y que gracias a ella estoy realizando este sueño: mi mamá Maribel.

A mi hermano Yoskiel, mi papá Mateo y mi abuelita Nieves por brindarme todo el amor del mundo.

A la nueva alegría de nuestra familia mi sobrinito Leandro Fabian por llenarnos el corazón de momentos felices en cada minuto. A mí cuñada Janet, a mis tíos y primos que siempre han estado para compartir cada momento junto de felicidad.

Al amor de mi vida Javy por estar siempre presente para mí en estos 5 años de duro batallar, por darme todo su apoyo, amor y ayuda incondicional, por compartir conmigo y hacer este sueño realidad, gracias por todo mi amor.

A mis suegros Nancy y Piro por venir desde tan lejos a compartir conmigo este momento tan importante de mi vida.

A mi amigo y compañero de tesis Eric por toda su dedicación y excelente trabajo.

A la Revolución y a Fidel por ser autores de tan bella obra.

De Eric

A mi mamá, que aunque por ley de la vida ya no está conmigo, gracias a ella estoy aquí. Todo mi amor para ti mamí donde quiera que estés, que dios te tenga en la gloria.

A mi papá, que siempre ha estado a mi lado y me ha apoyado, gracias por todo, sin ti no hubiera pasado esta etapa de mi vida, te quiero.

A mi hermano y demás familia, por estar en mi vida y ser incondicionales en su apoyo.

A mi novia, amiga, compañera, amante, a mi amor, a mi todo: Liset, por compartir su vida todos estos años conmigo, por el apoyo en los momentos difíciles, por toda la alegría que hemos vivido, por ser ella mi vida.

A mi compañera de Tesis y amiga Yanet, por su dedicación y eficiencia para lograr nuestro objetivo.

A la Revolución y a Fidel, por crear esta universidad y permitirme ser un estudiante más.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de este trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2014.

Yanet Crespo Diaz

_____

Firma del Autor

Eric Rodríguez Ochoa

_____

Firma del Autor

Dr.C. Febe Angel Ciudad Ricardo

_____

Firma del Tutor

Dr.C. Amed Abel Leiva Mederos

_____

Firma del Tutor

Ing. Walfrido Serrano Pérez

_____

Firma del Tutor

Dr.C. Febe Angel Ciudad Ricardo

Graduado como Ingeniero Informático en el año 2004 por la Universidad de Holguín “Oscar Lucero Moya” (UHOLM) y el Instituto Superior Politécnico “José Antonio Echeverría” (CUJAE). Titulado como Máster en Informática Aplicada en el año 2007 por la Universidad de las Ciencias Informáticas (UCI) y obtuvo el grado científico de Doctor en Ciencias de la Educación – Especialidad Tecnología Educativa en el año 2012 por la Universidad de La Habana (UH). Imparte su docencia de pregrado en las disciplinas de Ingeniería y Gestión de Software, Metodología de la Investigación Científica y Formación Pedagógica. Es miembro de los claustros de las maestrías de Informática Aplicada, Informática Avanzada, Gestión de Proyectos y Educación a Distancia de la UCI. Desarrolla sus investigaciones en las temáticas de Ingeniería y Gestión de Software, con énfasis en el área del Software Educativo; así como en la Tecnología e Informática Educativas. Ha publicado diversos artículos científicos y ha participado en diferentes eventos nacionales e internacionales en estas áreas del conocimiento. Ha sido arquitecto, analista y líder de proyectos de desarrollo de software, jefe de departamento docente y asesor técnico – docente. Actualmente se desempeña como Director del Centro de Innovación y Calidad de la Educación (CICE) de la UCI.

Correo electrónico: fciudad@uci.cu

Dr.C. Amed Abel Leiva Mederos

Diplomado en Sistemas Inteligentes en la Universidad de Essex, Reino Unido.

Diplomando en Lingüística Computacional, Universidad Pompeu Fabra, Barcelona.

Profesor Auxiliar Facultad de Ingeniería Industrial UCLV.

Correo electrónico: amed@uclv.edu.cu

Ing. Walfrido Serrano Pérez

Trabaja en el CICE desde el 2009, imparte clases en la Facultad 1 desde entonces en las asignatura de Programación Web y el curso optativo Elementos de Hardware, trabaja además vinculado al proyecto de Gestión de Archivos de la misma facultad, ha participado como expositor de los productos de la universidad en los eventos internacionales Universidad 2010 y Universidad 2012, así como en el salón de exposición de la UCI. Trabaja en la administración del Sistema de Encuestas UCI apoyando el proceso de caracterización de los estudiantes. Ha tutorado varias tesis en los años anteriores. Ha obtenido siempre excelente en la evaluación profesoral.

Correo electrónico: wserrano@uci.cu

SÍNTESIS

Una de las insuficiencias que se presentan en la Universidad de las Ciencias Informáticas está relacionada con la Superación Pedagógica de sus docentes. Para dar solución a este tema, en la institución se elaboró una Estrategia de Superación Pedagógica del Claustro. Este resultado necesita como una de las entradas, los perfiles de los profesores; los cuales presentan dificultad en su elaboración, dado que la concepción del Sistema Automatizado que soporta la estrategia, realiza solamente un análisis cuantitativo de los datos obtenidos del Sistema de Encuestas. Es por ello, que la presente investigación tiene como objetivo, desarrollar un subsistema de análisis semántico de texto en el proceso de análisis de datos de la concepción del sistema automatizado que soporta la estrategia, que permita extraer información en los textos de las respuestas a las preguntas abiertas, para ser utilizadas en la confección de las estrategias individuales de Superación Pedagógica. El subsistema implementa un algoritmo de extracción de información estructurada de un texto, compuesto por 3 etapas. Este se desarrolla sobre el área de conocimiento Minería de Texto y utiliza una ontología como base de conocimiento. La solución propuesta permite obtener conocimiento previamente desconocido de las respuestas a las preguntas abiertas, y ser utilizado en la confección de las estrategias individuales de Superación Pedagógica de los docentes, contribuyendo a perfeccionar la concepción de la preparación docente de cada profesor.

Palabras clave: análisis semántico, información, minería de texto, ontología

INTRODUCCIÓN

Educar a una sociedad para la vida, constituye un complejo, arduo y difícil trabajo, al ser un deber fundamental para el hombre, y más al borde de una sociedad cada vez más compleja que necesita jóvenes mayormente preparados, conscientes, con ideales y valores bien definidos, siendo capaces de afrontar los retos del presente y del futuro con una identidad segura y propia de una buena cultura (Alizegui, 2013). Es por ello que la base del progreso para la sociedad reside en la Educación.

En Cuba, la educación se organiza mediante el **Sistema Nacional de Educación**, comprendido por cuatro niveles: Educación Preescolar, Educación Primaria, Educación General Media (comprende dos subniveles: la educación secundaria básica y preuniversitaria) y Educación Superior (UNESCO, 1999). Este último nivel ha desempeñado un papel protagónico en la conformación de la cultura y de la sociedad cubana actual. Es por esto, que la preparación de los docentes constituye una tarea de gran importancia y necesidad para preservar, enriquecer y potenciar los logros que ha tenido este nivel educacional en Cuba.

La superación pedagógica a los docentes es concebida por el Ministerio de Educación Superior mediante la Educación de Postgrado. Existen todavía un conjunto de deficiencias en esta área del conocimiento, detectadas en un análisis realizado a diferentes diseños de superación pedagógica en el país por (Ciudad, y otros, 2013), quienes constatan que existe:

- Diferencias en el reconocimiento de necesidades de superación en el área pedagógica.
- Variedad de contextos en que se desarrolla la preparación pedagógica.
- Diversidad de objetivos y contenidos declarados en el área pedagógica para organizar la superación profesional y la formación académica.

La situación descrita no excluye a la Universidad de las Ciencias Informáticas (UCI), sino que en su contexto actúan condiciones similares, declaradas en el Informe final de evaluación institucional del MES a la UCI, disponibles en (IFAI-UCI-MES, 2010). En este informe, se señalaron entre otros aspectos, la necesidad de elevar la preparación pedagógica del claustro, donde se hace de suma importancia atender este asunto en la Universidad.

La UCI cuenta con un Centro de Innovación y Calidad de la Educación (CICE) en el cual se elaboró la Estrategia de Superación Pedagógica del Claustro UCI (ESPC-UCI), aprobada por el Consejo Universitario. Según (Ciudad, y otros, 2013), de acuerdo al diagnóstico preliminar realizado para la ESPC-UCI y para cumplir el objetivo previsto, se puede dividir el claustro de la UCI en cuatro grupos fundamentales:

- Los graduados de perfil pedagógico.
- Los graduados de perfil no pedagógico que han recibido preparación pedagógica.
- Los graduados de perfil no pedagógico que han recibido alguna preparación pedagógica.
- Los graduados de perfil no pedagógico que no han recibido preparación pedagógica.

Por lo cual dicha estrategia necesita como una de las entradas los perfiles de los profesores, que según (Sánchez & Jaimes, 1985), *«es el conjunto de roles, conocimientos, habilidades y destrezas, actitudes y valores que posee un recurso humano determinado para el desempeño de una profesión».* Elaborar los perfiles constituye sin dudas, un proceso que se dificulta actualmente, teniendo en cuenta que en la concepción del Sistema Automatizado de Superación Pedagógica (SASPED) se realiza solamente un análisis cuantitativo de los datos obtenidos del sistema de Encuestas – Lime Survey.

El sistema de Encuestas cuenta con dos tipos de preguntas: preguntas cerradas y preguntas abiertas. El tipo de preguntas cerradas consiste en que el encuestado debe limitarse en su respuesta a una de las alternativas provistas, sin posibilidad de generar mayor información. Esto facilita grandemente al uso de métodos estadísticos, lo que permite realizar un análisis cuantitativo. Sin embargo, las preguntas abiertas le permiten al encuestado tener una mayor amplitud para responder a cada una de estas, pero a las respuestas generadas no se le realiza ningún tipo de análisis, sino que esta información se almacena de forma textual para su posterior consulta y posible análisis por un especialista humano.

Realizar el análisis cualitativo es de gran importancia, según (Reyes, 2012), *«el propósito de analizar los datos es el de articular y estructurar éstos para describir las experiencias de las personas bajo su propia óptica, lenguaje y forma de expresarse, interpretando y evaluando unidades, categorías y patrones, para dar sentido a los datos dentro del marco del planteamiento del problema»*. Lo anterior, le permite al investigador capturar la perspectiva del encuestado y llevar a cabo un análisis más cercano a lo que expresa el mismo.

Con un análisis de este tipo se podría extraer información de los textos de las respuestas a las preguntas abiertas de la encuesta de Superación Pedagógica, que según (Chiavenato, 2006, pág. 110), información *«es un conjunto de datos con un significado, o sea, que reduce la incertidumbre o que aumenta el conocimiento de algo. En verdad, la información es un mensaje con significado en un determinado contexto, disponible para uso inmediato y que proporciona orientación a las acciones por el hecho de reducir el margen de incertidumbre con respecto a nuestras decisiones».*

Por lo antes expuesto, se identifica que en la concepción del sistema SASPED el proceso de análisis de datos se caracteriza por presentar:

- Preguntas cerradas, limitando la obtención de los datos e interfiriendo en la extracción de la información.
- Análisis de datos utilizando solo métodos estadísticos.
- Insuficiente diversificación en el análisis de los resultados de los diferentes tipos de preguntas.

En este sentido se identifica el siguiente **problema a resolver**: insuficiencias en el proceso de análisis de datos de la concepción de SASPED, limitan la extracción de información en un texto para ser utilizadas en la confección de las estrategias individuales de Superación Pedagógica.

Este problema se enmarca en el **objeto de estudio:** proceso de análisis de datos de la concepción de SASPED; donde el **campo de acción** lo comprenden las herramientas de análisis semántico de texto. Para resolver el problema identificado se propone el siguiente **objetivo general,** desarrollar un subsistema de análisis semántico de texto en el proceso de análisis de datos de la concepción de SASPED, que permita extraer información en los textos de las respuestas a las preguntas abiertas, para ser utilizadas en la confección de las estrategias individuales de Superación Pedagógica.

Para dar cumplimiento al objetivo general anteriormente planteado se definen las siguientes **tareas de investigación:**

1. Establecimiento de los fundamentos teóricos-metodológicos del proceso de análisis de datos y las herramientas de análisis semántico de texto.
2. Caracterización del proceso de análisis de datos obtenido de la concepción de SASPED.
3. Desarrollo del subsistema informático de análisis semántico de texto para SASPED.
4. Validación de la contribución lograda a través de la introducción del subsistema de análisis semántico de texto en el proceso de análisis de datos de la concepción de SASPED, en lo referente a la extracción de información en los textos de las respuestas a las preguntas abiertas.

La **hipótesis** a defender en la investigación plantea que: la utilización de un subsistema de análisis semántico de texto en el proceso de análisis de datos de la concepción de SASPED, permitirá extraer información en los textos de las respuestas a las preguntas abiertas, para ser utilizadas en la confección de las estrategias individuales de Superación Pedagógica.

Tomando como base lo planteado por (Hernández, Fernández-Collado, & Baptista, 2006), la **población** está conformada por todas las respuestas a la pregunta abierta del diagnóstico de la ESPC-UCI y se identifica como **unidad de análisis** la respuesta a la pregunta abierta. El **tipo de muestra** es probabilística y se utiliza la **técnica de muestreo** aleatoria simple. Para conformar la muestra se trabajó con un error estándar del 5% y un nivel de confianza del 95%. Con una población de 15 respuestas, se necesitan 6 para integrar la muestra.

En el desarrollo de la investigación se utilizaron un conjunto de **métodos científicos**, todos bajo la concepción dialéctico – materialista como método general.

Como parte de los **métodos teóricos** utilizados se encuentran:

- **Histórico – lógico:** para la determinación de los antecedentes en su devenir histórico, tendencias y regularidades del objeto de estudio y el campo de acción.
- **Hipotético – Deductivo:** para arribar a nuevos conocimientos y predicciones, que posteriormente son sometidos a verificaciones empíricas.

- **Análisis – síntesis e Inducción – deducción:** para la determinación de las generalidades y especificidades en el objeto de estudio y el campo de acción; así como en la fundamentación teórica y elaboración del subsistema de análisis semántico de texto.
- **Modelación:** se utilizó con el objetivo de generar los artefactos necesarios para estudiar y transformar el proceso de análisis de datos de la concepción de SASPED.

Se utilizaron como **métodos empíricos**:

- **Observación:** con el objetivo de obtener una información precisa y real del fenómeno que se estudia, en este caso, el proceso de análisis de datos de la concepción de SASPED.
- **Análisis documental:** se utilizó este método con el objetivo de obtener información mediante la recolección y selección de documentos relacionados con el tema que se investiga, con la finalidad de obtener resultados que pudiesen ser base para el desarrollo de la creación científica.
- **Análisis de contenido:** se utilizó este método con el objetivo de analizar y valorar la información obtenida y de esta forma descubrir una serie de ideas, que no están explícitas como tales en el texto, sino que se obtienen tras un proceso de abstracción y de elaboración.
- **Experimento:** esta actividad científica se dirige a comprobar la validez de la hipótesis planteada. Además, modificará de forma controlada las condiciones en que ocurre el objeto de estudio que se investiga para medir el comportamiento de las variables de investigación.

El trabajo de diploma consta de introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, glosario de términos y anexos. En el primer capítulo se presentan los fundamentos teórico – metodológicos del objeto de estudio y el campo de acción. En el segundo se muestra la caracterización del objeto de estudio y el análisis y diseño de la propuesta del subsistema de análisis semántico de texto. En el tercero se muestra la implementación y se valida el subsistema, en donde se analizan los resultados de su introducción en la práctica.

CAPÍTULO 1: ANALIZADOR SEMÁNTICO: FUNDAMENTOS TEÓRICOS-METODOLÓGICOS PARA SU DESARROLLO

En el presente capítulo se abordan los principales conceptos relacionados con la problemática a resolver. Se exponen las definiciones asociadas con el proceso de análisis de datos, las herramientas de análisis semántico de texto, y la propuesta del algoritmo de extracción de información estructurada basado en ontología. Se realiza un estudio científico del tema que se aborda para conformar el basamento teórico utilizado en la solución del problema científico. Además, se definen las herramientas, metodología y lenguaje a utilizar en la solución propuesta.

1.1. Conceptos relacionados con el proceso de análisis de datos

Para una mayor comprensión del presente trabajo investigativo, se analizan los conceptos asociados al proceso de análisis de datos, definido como objeto de estudio de la investigación.

El Diccionario de la Real Academia Española (DRAE, 2014) define el concepto de "proceso" de la siguiente manera:

Proceso: *«conjunto de las fases sucesivas de un fenómeno natural o de operación artificial*».

Por su parte (Mira, Gómez, Blaya, & García, 2006) expresan que un proceso es un *«conjunto de actuaciones, decisiones, actividades y tareas que se encadenan de forma secuencial y ordenada para conseguir un resultado que satisfaga plenamente los requerimientos del cliente al que va dirigido».*

Cuando se habla de proceso de análisis de datos los autores (Rodríguez, Gil, & García, 1996) lo definen como un *«conjunto de manipulaciones, transformaciones, operaciones, reflexiones, y comprobaciones que se realizan con el fin de extraer significados relevantes en relación con un problema de investigación».*

Para (Hernández, García-Sanz, & Maquilón, 2009) el análisis de datos *«pretende examinar las partes de algo por separado, intentando conocer las relaciones existentes entre cada una de estas partes, con la intención de reconstruir el significado global».*

Según (Mejía, 2011), el análisis de datos cualitativos *«se centra en los sujetos, su objetivo es comprender a las personas en su contexto social. El criterio del análisis es de tipo holístico, en el sentido de que se observa y estudia a los individuos en todas las dimensiones de su realidad. En cambio, el análisis de datos cuantitativos privilegia sólo las variables y sus relaciones, enfatizando las dimensiones aisladas de los fenómenos sociales».*

A su vez, los autores (Rodríguez, Lorenzo, & Herrera, 2005) opinan que el análisis de datos cualitativos es el *«proceso mediante el cual se organiza y manipula la información recogida por los investigadores para establecer relaciones, interpretar, extraer significados y conclusiones. El análisis de datos cualitativos se caracteriza, por su forma cíclica y circular, frente a la posición lineal que adopta el análisis de datos cuantitativos».*

Para (Hernández, Fernández-Collado, & Baptista, 2006), *«el proceso cuantitativo es secuencial y probatorio. Cada etapa precede a la siguiente y no se puede brincar ni eludir pasos, aunque se puede*

redefinir alguna fase. El proceso cualitativo es en espiral o circular, las etapas a realizar interactúan entre sí y no siguen una secuencia rigurosa. A su vez el análisis de datos cualitativos se caracteriza por ser eclético, paulatino y paralelo al muestreo y a la recolección de datos, distinguiéndose del análisis cuantitativo por no seguir reglas ni procedimientos concretos, ya que es el investigador quien construye su propio análisis»; coincidiendo con esta definición los autores (Reyes, 2012), y (Carrillo, Leyva-Moral, & Medina, 2011),

Por su parte, (Maduro & Rodríguez, 2008) concluyen en sus estudios que *«el análisis de datos cualitativos ha llegado a ser un método científico capaz de ofrecer interpretaciones a partir de datos esencialmente verbales, simbólicos o comunicativos»*. Además, citan que *«las principales finalidades del análisis cualitativo según expresan algunos autores son: la búsqueda del significado de los fenómenos a partir de los datos concretos, confirmar o rechazar hipótesis, y ampliar la comprensión de la realidad como una totalidad».*

Varios autores, como (Rodríguez, Gil, & García, 1996), (Rodríguez, Lorenzo, & Herrera, 2005), (Mayz, 2009) y (Mejía, 2011) en sus investigaciones definen una serie de fases o grandes tareas para realizar el análisis de datos, debido a que no existe un modo único o estandarizado de llevar a cabo el análisis, pues cada autor lo adapta a su objeto de estudio con la finalidad de ofrecer una herramienta útil y comprensiva.

Teniendo en cuenta el análisis realizado, los autores de este trabajo concluyen que el proceso de análisis de datos constituye una de las tareas más significativas dentro del proceso de investigación, pues a través del análisis se puede arribar a resultados y conclusiones. Esto permite profundizar en el conocimiento de la realidad que se investiga, ya sea por el análisis de datos cualitativos o cuantitativos. Aunque existen diversas formas de realizar el análisis de datos, existe un punto único de acuerdo entre los investigadores sobre la idea de que el análisis es el proceso de extraer sentido a los datos.

1.2. Conceptos relacionados con el análisis semántico de texto

Cuando se desea conocer el significado de un texto y captar el mensaje que este ofrece, se necesita que la cadena de caracteres transite por diferentes fases de procesamiento.

Para la comprensión del texto, es necesario definir, delimitar y clasificar las unidades que componen las palabras, y así determinar las categorías gramaticales (sustantivos, verbos, adjetivos, artículos, entre otros). Esta fase denominada **morfología**, según el (DRAE, 2014), es la *«parte de la gramática que se ocupa de la estructura de las palabras»*.

A partir de la identificación de la categoría de las palabras, es necesario establecer la relación entre ellas dentro de una oración. Es decir, identificar el rol de la palabra dentro de la frase y las dependencias con las otras palabras. Esta etapa identificada como **sintaxis**, según el (DRAE, 2014), es la *«parte de la gramática que enseña a coordinar y unir las palabras para formar las oraciones y expresar conceptos»*.

Una vez reconocida las expresiones sintácticas del texto, es permisible el estudio de su significado, es decir, establecer relaciones entre significados y significantes. Identificándose finalmente la **semántica**, definida por (DRAE, 2014) como lo *«perteneciente o relativo a la significación de las palabras. Estudio del significado de los signos lingüísticos y de sus combinaciones, desde un punto de vista sincrónico o diacrónico».*

Cuando se habla de análisis semántico (Aguilar, 2009) lo define como un *«mapeo desde las expresiones del lenguaje hasta entidades mentales o cognitivas».* Para (Piñero, 2010), *«el análisis semántico investiga el significado de cada uno de los "lexemas" del texto, es decir, los vocablos que tienen un significado independiente y por sí mismo, intentando establecer qué semas (unidades de significado en el interior de un lexema) pueden hallarse contenidos en ellos».*

Por su parte, (Kutschera, 1979), define la semántica como la ciencia de los significados de los signos lingüísticos o de los enunciados orales o escritos. Este término se deriva del griego sema o sémeion (signo). En la lingüística moderna, "sema" es la unidad mínima de significado, llamada también rasgo o componente semántico. Se ocupa de la relación entre la forma y el contenido, entre lo significante y lo significado en las palabras, en las frases y en los textos. El análisis semántico de un texto intenta responder a la pregunta ¿qué quiere decir un texto y qué establece lo que significan determinadas expresiones y frases utilizadas en un texto?

Varios autores como (Kutschera, 1979) y (Piñero, 2010) plantean que el análisis semántico investiga el significado de cada uno de los "lexemas" del texto, se ocupa de la relación entre lo significante y lo significado en las palabras, en las frases y en los textos. En la presente investigación se adopta la posición de los últimos autores mencionados y se concluye que, a través del análisis semántico se descubre que expresa un texto, mediante el significado de los lexemas, de las frases y de las expresiones.

1.3. Herramientas existentes para el análisis semántico de texto

En el extranjero, han sido elaboradas y diseñadas varias soluciones informáticas, con la finalidad de realizar análisis semántico de texto. A continuación se muestran algunas de estas herramientas:

1. **El Manchador de Textos (EMT)** es una herramienta computacional, implementada en el contexto del Proyecto FONDECYT[1], que permite buscar determinados rasgos lingüísticos en un conjunto de textos almacenados en formatos electrónicos y agrupados con el fin de estudiar una lengua determinada. Los resultados obtenidos en esta búsqueda se presentan a través del coloreado de las palabras o estructuras lingüísticas que han sido buscadas. Además, el programa permite calcular un índice que da cuenta de la aparición conjunta de los rasgos en el texto.
   Disponible en: < http://www.elgrial.cl/manchador1.html >

[1] Fondo Nacional de Desarrollo Científico y Tecnológico del Ministerio de Educación de Chile.

2. **Tropes Zoom** es un software de indexación y búsqueda en lenguaje natural y de análisis documental basado en la comprensión de los contenidos a tratar. Cada palabra significativa se inscribe en una cadena de equivalentes semánticos. Tropes Zoom dispone de múltiples herramientas de análisis, entre las cuales cabe destacar:
   - Búsqueda por criterios semánticos, con resolución de las ambigüedades.
   - Estructuración de la información por clasificación automática de los ficheros hallados.

   Tropes Zoom es una edición limitada privativa y se ejecuta solamente en Windows.
   Disponible en: < http://www.semantic-knowledge.com/zoom.htm >
3. **TextAnalyst** es una herramienta de análisis de contenido textual para el análisis semántico, la navegación y la búsqueda de los textos no estructurados. Ayuda a resumir rápidamente y agrupar documentos, además de brindar la posibilidad de crear una "red semántica" del contenido del texto. Se pueden obtener los párrafos del texto que se encuentran ligados con cada uno de los nodos de la red semántica. El sistema determina qué conceptos –palabras o combinaciones de palabras- son las más importantes en el contexto del texto bajo estudio. Es una herramienta privativa y se ejecuta solamente en Windows, dirigida únicamente a textos en idioma inglés.
   Disponible en: < http://www.megaputer.com/textanalyst.php >
4. **T-LAB** es un software compuesto por un conjunto de herramientas lingüísticas y estadísticas para el análisis de contenido, el análisis del discurso y la minería de textos. Utiliza métodos automáticos y semi-automáticos que permiten descubrir rápidamente relaciones significativas entre palabras, temas y variables. Está diseñado para la plataforma Windows y es privativo.
   Disponible en: < http://www.tlab.it/es/presentation.php >
5. **WordSmith Tools** es una herramienta distribuida por Oxford University Press y que permite explotar grandes conjuntos de textos mediante búsquedas basadas en parámetros contextuales o estadísticos. Es una herramienta de análisis de corpus monolingüe para Microsoft Windows que, a base de archivos en formato plano, permite crear concordancias, listas de frecuencia y listas de palabras clave que aparecen con frecuencia en el texto.
   Disponible en: < http://www.lexically.net/wordsmith/ >
6. **Concordance** es una herramienta enfocada en la concordancia lógico-semántica que muestra cada palabra en su contexto y utiliza textos de cualquier tamaño, limitado sólo por el espacio disponible en el disco y la memoria. Permite contar palabras, hacer listas de palabras, listas de frecuencia de palabras, e índices. Trabaja con diferentes lenguas y es soportado por el sistema operativo Windows. Disponible en: < http://www.concordancesoftware.co.uk/features.htm >
7. **Textquest** es una herramienta para el análisis de textos. Ofrece una variedad de análisis, desde una simple lista de palabras a un análisis de contenido o el análisis de la legibilidad. Permite

realizar búsquedas a partir de patrones de consulta basados en palabras. El programa funciona aplicando etiquetas de categorías, lo que obliga a documentar con detalles todas las categorías usadas. La herramienta está diseñada para las plataformas Windows y Mac OS-X, dirigida a textos en idioma inglés y alemán. Disponible en: <http://www.textquest.de/ >

A nivel nacional se llevó a cabo una investigación en el año 2011 para la realización de una herramienta de análisis semántico, elaborándose para esto el algoritmo descrito a continuación:

Algoritmo GRIAL-WSD: Es un algoritmo basado en el conocimiento propuesto por (Fernández, 2011) en su tesis de Maestría para la desambiguación del sentido de las palabras (Word Sense Disambiguation: WSD).

El algoritmo integra varios métodos mediante determinadas medidas estadísticas, basándose en una base de datos léxica y el algoritmo propuesto por (Lesk, 1986), con diferentes adaptaciones realizadas a éste por parte de (Banerjee, 2002), sirviendo esto como base para la elaboración del algoritmo. La herramienta que contiene este algoritmo no está disponible para la comunidad científica.

Las herramientas para el análisis semántico analizadas anteriormente, permiten el pre-procesamiento de texto, la búsqueda de rasgos lingüísticos, la creación de concordancia, la creación de listas de frecuencia, la identificación de palabras clave, las relaciones significativas entre palabras y la resolución de las ambigüedades. Sin embargo estas herramientas no dan solución al problema de la presente investigación ya que presentan las siguientes limitaciones: no utilizan ontologías como base de conocimiento, están dirigidos al análisis de textos en idioma extranjero, no extraen información implícita del texto, están basadas en software propietario y no son multiplataforma. Además, no cuentan con funcionalidades que realicen el análisis semántico de texto específico al dominio de las encuestas realizadas a los profesores, para la elaboración de sus perfiles. Por lo antes descrito, se presenta una propuesta de subsistema de análisis semántico de texto, el cual implementa un algoritmo de extracción de información de un texto a partir de un análisis lingüístico y utiliza como base de conocimiento una ontología de dominio. Esto permite mejorar el proceso de análisis de datos y la extracción de información explícita e implícita en las respuestas a las preguntas abiertas, la cual genera conocimiento previamente desconocido que es de importancia para la confección de las estrategias individuales de Superación Pedagógica.

1.4. Algoritmos de procesamiento de textos

La gran cantidad de información, textual y no estructurada, que se encuentra almacenada y que continuamente se genera, demanda de iniciativas que propicien un mayor aprovechamiento de la información para el descubrimiento de conocimiento y la toma de decisiones.

Bajo este principio se han desarrollado varias herramientas, todas encaminadas a extraer información desconocida de textos no estructurados para dar respuestas a sus problemáticas.

Ejemplos de esto, se encuentran en trabajos como los de (Rodríguez & Simón, 2013), los cuales desarrollaron una herramienta, donde la información extraída es estructurada en forma de grafo, específicamente mediante un Mapa Conceptual (MC). Estos autores definen que en el proceso de extracción y estructuración de información se parte de una información textual (contenida en un fichero o introducida manualmente) en lenguaje natural no estructurado, y se ejecutan un conjunto de tareas que pueden ser agrupadas en tres etapas: pre-procesamiento, extracción de información (frases conceptuales y relaciones) y refinado y construcción del Mapa Conceptual. En la figura uno se muestra el esquema general del método.

Además, se puede citar a los autores (Kowata, Cury, & Beores, 2010), los cuales presentan un método de construcción de Mapa Conceptual a partir de texto en el que se incluyen las siguientes tareas:

1. Extracción de Texto Plano
2. Segmentación del Texto
3. Extracción de tokens[2]
4. POS Tagging (Etiquetado gramatical).
5. Reconocimiento de elementos Centrales Candidatos
6. Intérprete de dependencias
7. Constructor del MC

En esta propuesta se consideran las frases sustantivas, verbales y preposicionales como primeros candidatos a ser elementos principales de los Mapas Conceptuales.

Por otra parte, la elaboración automática de resúmenes cobra vital importancia, pues a través del procesamiento de texto y la lingüística computacional se puede extraer información relevante de un documento.

Autores como (García & Medina, 2010), en su investigación plantean que independientemente del nivel al que se procese la información para obtener el resumen, existen varios tipos de pre-procesamientos que se les realiza a los textos, ya sea para obtener un conjunto de términos más representativos de los documentos, reducir dimensionalidad, entre otros.

Entre las **operaciones más comunes realizadas en el pre-procesamiento** inicial de los documentos se tienen:

- Eliminación de signos de puntuación, espaciados, acentos, reducción de mayúsculas, entre otros.
- Eliminación de las palabras vacías (stopwords). Las stopwords son palabras que carecen de significado a la hora de representar un documento, ya que con seguridad aparecen en todos los documentos de la colección. Se trata de palabras como artículos, adverbios, pronombres, preposiciones, entre otros.
- Extracción de raíces o de lemas (lematización).
- Etiquetado gramatical.
- Extracción de entidades nombradas.

[2] Partes de una oración (palabras, números, signos de puntuación, etc.)

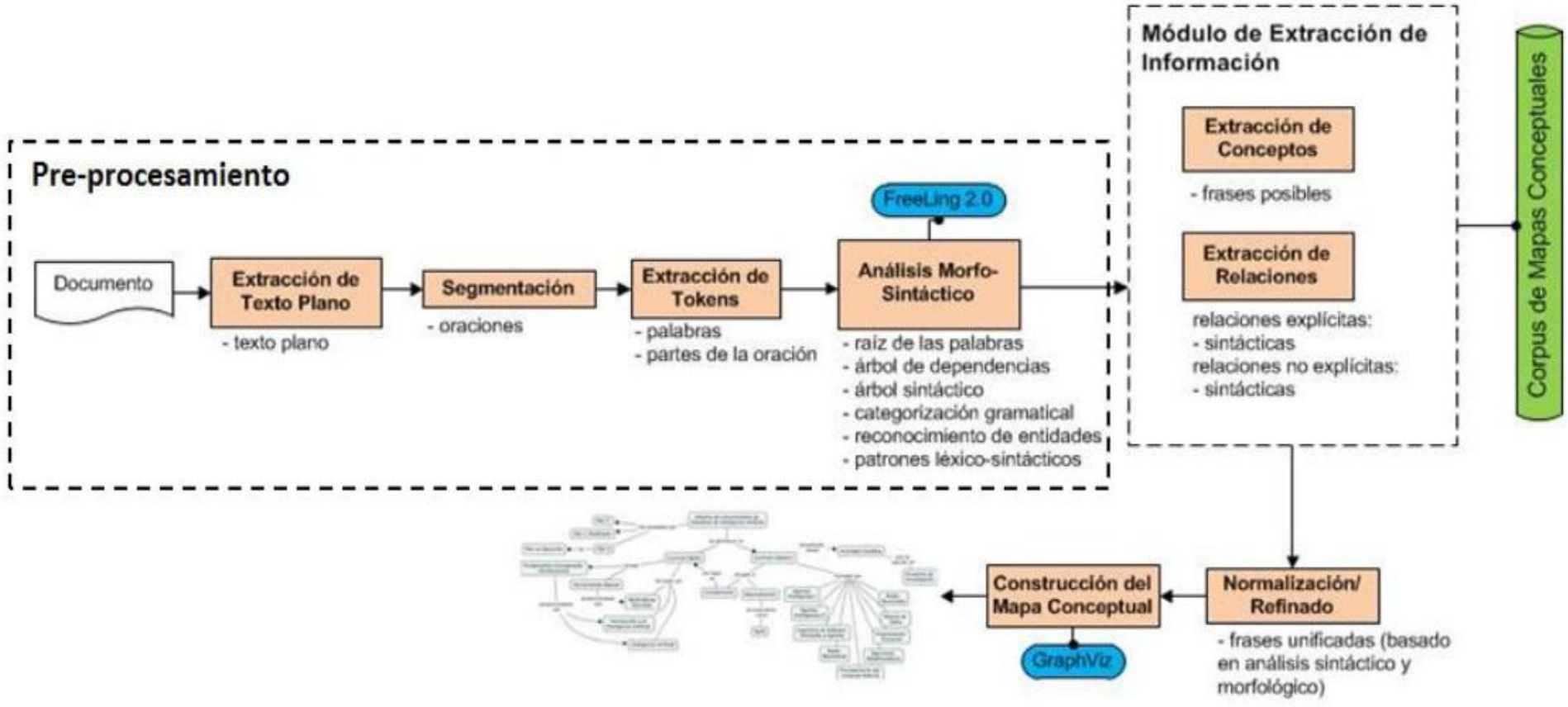


Figura 1: Esquema Funcional del Método de Extracción de Información Estructurada en forma de MC [Tomado de (Rodríguez & Simón, 2013)]

1.5. Propuesta del algoritmo de extracción de información estructurada de un texto

El algoritmo que se propone, toma como base las propuestas de las investigaciones anteriores, dado que las mismas están encaminadas al análisis y procesamiento de textos. Este algoritmo procesa las respuestas a las preguntas abiertas de las encuestas de Superación Pedagógica. Las respuestas se encuentran almacenadas en lenguaje natural, lo que limita su procesamiento computacional. Realizar este procesamiento permitirá descubrir conocimiento, lo que facilitaría la toma de decisiones a la hora de trazarle la Estrategia Individual de Superación Pedagógica a cada docente. La **Minería de Texto** (MT) constituye el área de conocimiento en la cual se desarrolló el algoritmo, debido a que en esta área, según (Feldman, y otros, 1998), se generan soluciones para el *«descubrimiento de conocimientos potencialmente útiles, y no explícito, en una colección de textos, a partir de la identificación y exploración de patrones interesantes»*.

Para (Kodratoff, 1999) *«la MT es la más reciente área de investigación del procesamiento de textos. Ella se define como el proceso de descubrimiento de patrones interesantes y nuevos conocimientos en una colección de textos, es decir, la minería de texto es el proceso encargado del descubrimiento de conocimientos que no existían explícitamente en ningún texto de la colección, pero que surgen de relacionar el contenido de varios de ellos»*.

Según (Hearst, 1999), *«la MT adopta un enfoque semiautomático, estableciendo un equilibrio entre el análisis humano y automático: antes de la etapa de descubrimiento de conocimiento es necesario procesar de forma automática la información disponible en grandes colecciones documentales y transformarla en un formato que facilite su comprensión y análisis. El procesamiento de grandes volúmenes de texto libre no-estructurado para extraer conocimiento requiere la aplicación de una serie de técnicas de análisis ya utilizadas en la Recuperación de Información (RI), el Procesamiento del Lenguaje Natural (PLN) y la Extracción de Información (EI)»*.

Las **técnicas** en la MT según (Botta & Cabrera, 2007) se estructuran básicamente en tres etapas:

- **Etapa de pre-procesamiento**: Es el proceso mediante el cual los textos se transforman en algún tipo de representación estructurada que facilite su análisis.
- **Etapa de representación**: La representación depende de la técnica de pre-procesamiento utilizada y determinará cuál será el algoritmo a utilizar en la etapa de descubrimiento.
- **Etapa de descubrimiento**: es el proceso en el cual a partir de una representación estructurada de la información, se descubre regularidades en los textos.

Para el desarrollo de estas etapas se pueden seguir varias **estrategias de la MT:** (Ver figura dos)

Etapa de pre-procesamiento	Tipo de representación	Tipo de descubrimientos
Categorización	Vector de temas	Nivel temático
Full-text	Secuencia de palabras	Patrones de lenguaje
Extracción de información	Tabla de datos	Relaciones entre entidades

Figura 2: Estado del arte de la minería de texto [Tomado de (Montes y Gómez, 2001)]

Como se puede comprender, todas las etapas están muy interrelacionadas, (Montes y Gómez, 2001) plantean que *«dependiendo del tipo de métodos usados en la etapa de pre-procesamiento es el tipo de representación del contenido de los textos construida; y dependiendo de esta representación, es el tipo de patrones descubiertos»*.

Teniendo en cuenta lo planteado por el último autor, se hace necesario para el desarrollo del algoritmo, seleccionar una de estas estrategias, por lo que se definieron una serie de indicadores a evaluar, siendo estos: unidad de análisis, tipo de representación intermedia y enfoque de descubrimiento. (Ver tabla uno)

Tabla 1: Indicadores analizados para la selección de la estrategia

Indicadores / Estrategia	Unidad de análisis	Tipo de representación intermedia	Enfoque de descubrimiento
Categorización	Colecciones de textos	A nivel documento	A nivel representación
Full-Text	Texto íntegro	A nivel documento	A nivel texto
Extracción de información	Palabras	A nivel concepto	A nivel Mundo

Una vez analizados estos indicadores se determinó que el algoritmo que se propone está guiado por la tercera estrategia, la cual está compuesta por: extracción de información, tablas de datos y relaciones entre entidades. La selección de la estrategia está sustentada porque en la solución del problema no es necesario analizar colecciones de textos, ni el texto completo, sino palabras, constituyendo esta la unidad de análisis.

Por otra parte, el tipo de representación intermedia que se obtiene es a nivel concepto, dado que se identifican objetos, temas o conceptos interesantes para el dominio específico de aplicación. A su vez, el enfoque de descubrimiento es a nivel mundo, pues lo que se necesita para resolver el problema es

descubrir relaciones entre conceptos, no siendo necesario descubrir patrones de lenguaje como sucede a nivel texto, ni representar los textos en un nivel temático como acontece a nivel representación.

Siguiendo esta estrategia se cumple con el objetivo de este algoritmo, el cual consiste en extraer la información significativa y esencial del texto, lo que permite descubrir relaciones entre conceptos y con esto a su vez información y conocimiento implícito que es de gran importancia para la toma de decisiones. Es por ello, que la minería de texto es la razón que fundamenta la propuesta de este algoritmo como perspectiva metodológica para la realización del análisis semántico de texto, a las respuestas de las preguntas abiertas de las encuestas de Superación Pedagógica.

Etapas del Algoritmo propuesto

De forma general, el algoritmo parte de una información textual en lenguaje natural no estructurado, y se ejecutan un conjunto de tareas que son agrupadas en tres etapas: ***pre-procesamiento***, ***descubrimiento*** (extracción de conceptos y sus relaciones) y ***obtención de resultados***. En la figura tres se muestra el esquema general del algoritmo.

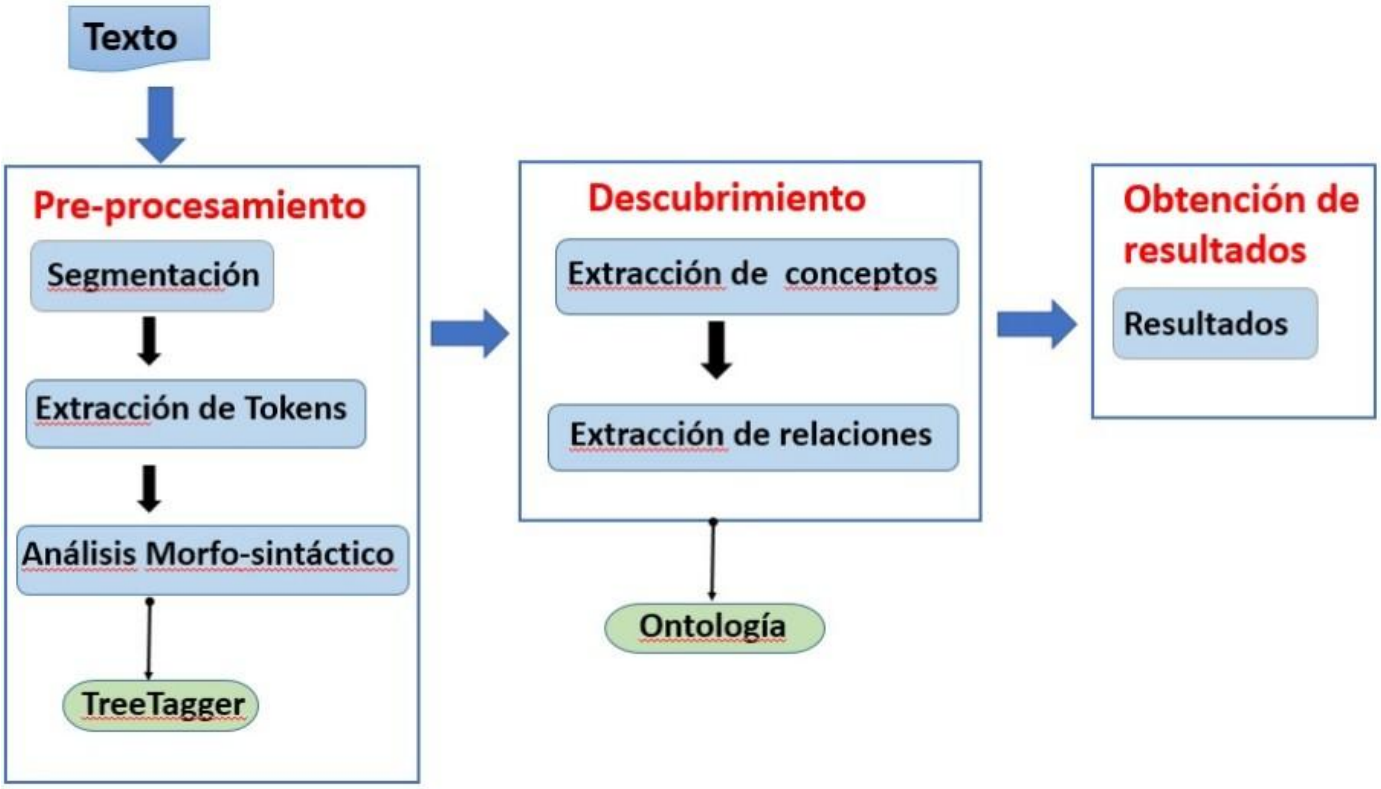


Figura 3: Esquema general del algoritmo propuesto

A continuación, se explican las etapas del análisis llevado a cabo en el algoritmo que se muestra en la figura anterior.

Pre-Procesamiento del texto

Segmentación de Texto: La segmentación de texto consiste en desfragmentar el mismo en párrafos y oraciones, las cuales se segmentan con la utilización de un algoritmo para determinar sus fronteras, a partir de la función de los puntos en el texto. De esta forma, se obtiene una lista de oraciones a ser procesada, como resultado de la segmentación.

Extracción de tokens: Este proceso divide cada oración en un conjunto de tokens, los cuales constituirán la base para análisis posteriores. Los tokens se identifican en una oración mediante un algoritmo, a partir de las fronteras entre las diferentes clasificaciones de tokens.

Análisis morfo-sintáctico: El análisis morfo-sintáctico es reconocer en una oración cada una de las partes que la componen. Indica de cada palabra que parte y función cumple en la oración. En el algoritmo el análisis morfo-sintáctico del texto se realiza a cada oración por separado. Inicialmente se etiqueta cada token con lo que se determina su raíz morfológica y categoría gramatical. Contar con los lemas permite establecer la forma estándar de las palabras, al igual que en un diccionario. Además, obtener la categoría gramatical de los términos contenidos en el texto posibilitará procesar solamente los sustantivos y los adjetivos, considerándose estos como contenedores de la mayor información que se desea transmitir en una oración.

Se realiza el Análisis morfo-sintáctico con la utilización de la herramienta de etiquetado **TreeTagger**. Esta herramienta recibe como entrada un texto con su correspondiente idioma y devuelve una lista con todas la entidades sintácticas contenidas en el texto, acompañada cada una de ella por su categoría gramatical y raíz morfológica.

Descubrimiento

La etapa de descubrimiento representa el núcleo del algoritmo propuesto, teniendo en cuenta que es donde se extrae información clave para la obtención de los resultados. En esta etapa se utiliza toda la información obtenida en la fase anterior y se dispone de una ontología como base de conocimiento.

Extracción de conceptos: El proceso de extracción de conceptos consiste en identificar aquellas palabras que pueden tener un sentido conceptual. Para la identificación de estos conceptos potenciales se utiliza la base de conocimiento, encontrándose en esta, las palabras claves referentes a un dominio específico.

Extracción de relaciones: La extracción de relaciones va dirigida a dos caminos fundamentales, la extracción de relaciones explícitas y la extracción de relaciones implícitas, que permiten enlazar a uno o más conceptos con otros, identificados estos en la fase anterior.

Obtención de los resultados

Esta etapa constituye la última del algoritmo. Luego de haber extraído los conceptos y sus relaciones, es posible obtener un resultado en forma de texto, compuesto por la información explícita e implícita del texto inicial.

Base de conocimiento: ontologías

Tal y como se explica en la etapa de descubrimiento, para el desarrollo del algoritmo, es necesario como base de conocimiento una ontología. A continuación se presenta la definición de la misma, así como la fundamentación de su selección como base de conocimiento en la MT.

El concepto de Ontología, según (Ramos & Nuñez, 2007) *«tiene su origen en la Filosofía, disciplina que trata de dar una explicación sistemática de la existencia; proviene de la conjunción de los términos griegos "ontos" y "logos" que significan existencia y estudio, respectivamente».*

El término ontología es adoptado por la inteligencia artificial a finales de la década de los 80 del Siglo XX para compartir y reutilizar conocimiento, mientras que en la segunda mitad de los 90 de ese siglo se incorpora a la ingeniería web para la inclusión de descripciones semánticas explícitas de recursos (contenidos y servicios). En ambas disciplinas, según criterio de (Berners-Lee, Hendler, & Lassila, 2001), *«una ontología se materializa en un documento o un fichero que define formalmente las relaciones entre términos»*.

Otros autores como (Noy & McGuinness, 2005) plantean que *«una ontología define un vocabulario común para investigadores que necesitan compartir información en un dominio. Ella contiene definiciones de conceptos básicos y sus relaciones que pueden ser interpretadas por una máquina».*

Por otra parte, autores como (Samper, 2005), (Vallez, 2009), Mamani (Mamani, 2010), (Codina, & Pedraza, 2011), (Flores, 2011) y (Proenza & Pérez, 2012), expresan que la definición más breve, extendida y a la vez probablemente la más citada y aceptada, se debe a (Gruber, 1993), el cual define que «*una ontología es una especificación explícita de una conceptualización»*, luego extendida por (Borst, 1997) definiendo la ontología como *«una especificación formal de una conceptualización compartida»,* sistematizadas ambas en la siguiente definición de (Studer, Benjamins, & Fensel, 1998) «*una especificación formal y explícita de una conceptualización compartida»*.

A su vez, los autores (Simón, Rosete, & Ceccaroni, 2012), en su investigación concluyen *que «en sentido general, una ontología es la base del procesamiento semántico; e incluye una red de conceptos, relaciones y axiomas para representar, organizar y entender un dominio de conocimiento y proporciona el marco de referencia común para todas las aplicaciones en cierto entorno»*.

Analizadas estas definiciones, en el presente trabajo se adopta la posición de (Simón, Rosete, & Ceccaroni, 2012) y se concluye que, a través de las ontologías se hace posible una comprensión común

de la estructura de la información y compartición del conocimiento entre diferentes actores de un dominio determinado, como pueden ser personas, organizaciones y sistemas de software.

Ontologías en la Minería de Texto

En el área de conocimiento Minería de Texto se recurren a las bases de conocimientos con el objetivo de modelar y almacenar bajo forma digital un conjunto de conocimientos que puedan ser consultados o utilizados. En esta área se han realizado varios estudios que utilizan ontologías como base de conocimiento, ejemplo de esto se evidencia en las investigaciones de los autores (Hu, He, Ji, & Wang, 2004), (Ning & Shihan, 2006), (Zhang, Cheng, & Qu, 2007), (Hennig, Umbrath, & Wetzker, 2008) y (Leiva-Mederos, Domínguez-Fernández, & Senso, 2012); donde la misma es utilizada para la desambiguación del sentido de las palabras, la extracción y búsqueda de información. A su vez, (Marina, 2008) plantea en su investigación que, «*las Ontologías tienen diferentes aplicaciones, además de las ventajas obvias de organizar el conocimiento en un dominio determinado. Las Ontologías se utilizan para etiquetado formal de todo tipo de documentos o conjuntos de palabras, para mejorar las búsquedas, para encontrar nuevas relaciones entre elementos y para inferir conocimiento desconocido previamente. Además, sirve de base a numerosos algoritmos de Procesamiento del Lenguaje Natural (PNL) con objetivos como la extracción de información, extracción de relaciones, resúmenes automáticos de documentos, respuestas automatizadas a preguntas y, en general, a todo lo relacionado con la minería de texto».*

Tomando como base lo planteado por el autor (Marina, 2008) y al autor (Leiva-Mederos A. A., 2012) cuando plantea que «*las ontologías son una agrupación de palabras o términos que describen un campo de saber completo, por tanto ver las ontologías como una posición independiente de los procesos extractivos es errónea»; y* teniendo en cuenta que en la solución del problema se necesita extraer información explícita e implícita de un texto, se decide en la investigación utilizar como base de conocimiento la ontología, siendo la de mayor probabilidad de ofrecer resultados positivos para satisfacer estas necesidades.

1.6. Herramientas y tecnologías

En este epígrafe se tratan los conceptos relacionados con las herramientas, tecnologías y metodología que se consideraron para el proceso de desarrollo del Subsistema de Análisis Semántico.

Sistema de almacenamiento de ontología

Los sistemas de almacenamiento de ontologías permiten mantener las ontologías en bases de datos e ir añadiendo nueva información, los mismos son diversos y su selección está dado tras evaluar una serie de indicadores: lenguaje de programación, nivel de documentación, tipo de ontología que permite gestionar, lenguaje de consulta que soporta e integración con Sistemas de Base de Datos. Entre las herramientas analizadas se encuentran JENA, KAON API y RAP.

Como sistema de almacenamiento se decidió utilizar **JENA**, teniendo en cuenta que está desarrollado sobre el lenguaje de programación Java. Jena ofrece una amplia gama de funcionalidades para analizar, manipular, almacenar y consultar datos RDF. Permite gestionar ontologías de tipo RDF, DAML+OIL y OWL; y posee gran cantidad de documentación publicada acerca de sus funcionalidades, la cual se encuentra fundamentada y actualizada. Jena permite crear modelos persistentes en Base de Datos tales como MySQL, Oracle, PostgreSQL. Es gratuito y de código abierto.

Lenguaje de construcción de ontologías

Los autores (Ramos & Nuñez, 2007) plantean que los lenguajes de construcción de ontologías deben contemplar ciertos aspectos como: sintaxis bien definida, semántica específica, suficiente expresividad, fácilmente traducible entre los lenguajes ontológicos y permitir eficiencia para realizar razonamientos. A continuación se presentan algunos lenguajes para el desarrollo de ontologías:

RDF (Resource Description Framework, Marco de Descripción de Recursos)

RDF es una recomendación de la W3C[3] para representar metadatos en la Web. Proporciona un medio para agregar semántica a un documento sin referirse a su estructura. RDF es una infraestructura para la codificación, intercambio y reutilización de metadatos estructurados.

Debido a que RDF no define ningún vocabulario particular para la creación de los datos, era necesario un lenguaje que proporcionara las primitivas apropiadas. En consecuencia, se creó la especificación RDF Schema (RDFS). RDFS (o lenguaje de descripción de vocabulario RDF) amplía RDF con algunas primitivas básicas (basadas en marcos) para modelar ontologías, tales como clases, propiedades e instancias. Todo lo expresable en RDF, es expresable en sintaxis lineal de XML. RDF provee un modo estándar para representar metadatos en XML.

DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer, Agente de lenguaje de marcado DARPA + Capa de Inferencia de Ontología)

DAML+OIL es un lenguaje de marcado semántico para recursos Web. Es un estándar propuesto por la W3C para la representación de ontologías y metadatos. Fusión de DAML y OIL que amplía RDF(S) con primitivas de lógica de descripciones. DAML consiste en un formalismo que permite a los agentes de software interactuar entre ellos, soporta tipos de datos complejos y estos tipos de datos provienen de XML Schema.

OWL (Web Ontology Language, Lenguaje de Ontología Web)

OWL es un lenguaje de marcado semántico desarrollado por la W3C para publicar y compartir ontologías sobre el World Wide Web. Es una extensión del vocabulario de RDF y se deriva de DAML+OIL. OWL está

[3] El World Wide Web Consortium (W3C) es una comunidad internacional que desarrolla estándares que aseguran el crecimiento de la Web a largo plazo.

diseñado para ser utilizado por aplicaciones que necesitan procesar el contenido de la información en lugar de sólo presentarla a las personas. Este lenguaje tiene tres sublenguajes:

- **OWL Lite**: la versión más simple para los programadores principiantes. Permite la jerarquía de clasificación y las restricciones simples.
- **OWL DL:** está diseñado para aquellos usuarios que quieren la máxima expresividad conservando completitud computacional (se garantiza que todas las conclusiones sean computables), y resolubilidad (todos los cálculos se resolverán en un tiempo finito). OWL DL incluye todas las construcciones del lenguaje de OWL, pero sólo pueden ser usados bajo ciertas restricciones (por ejemplo, mientras una clase puede ser una subclase de otras muchas clases, una clase no puede ser una instancia de otra).
- **OWL Full**: Es el sub-lenguaje más expresivo, su intención es ser utilizado en situaciones donde una alta expresividad es más importante que la capacidad de garantizar la completitud computacional y la posibilidad.

SKOS (Simple Knowledge Organization System, Sistema Simple de Organización del Conocimiento)

SKOS es un vocabulario RDF para la representación de sistemas de organización del conocimiento semi-formales, tales como tesauros, taxonomías, esquemas de clasificación y listas de encabezamiento de materias. SKOS está basado en RDF por lo que dichas representaciones puedes ser legibles por máquinas e intercambiarse entre aplicaciones de software. SKOS también proporciona un lenguaje conceptual de modelado muy sencillo e intuitivo para desarrollar y compartir nuevos sistemas de organización. Puede utilizarse independiente, o en combinación con lenguajes más formales, como el Lenguaje de Ontologías Web (OWL) para expresar formalmente estructuras de conocimiento sobre un dominio concreto ya que SKOS no puede realizar esta función al no tratarse de un lenguaje para la representación de conocimiento formal.

El lenguaje de construcción de ontologías **OWL** se estructura en capas debido a su complejidad y puede ser adaptado a las necesidades de cada usuario, al nivel de expresividad que se precise y a los distintos tipos de aplicaciones existentes. Debido a las características del lenguaje y al grado de expresividad de los elementos y sus relaciones, es seleccionado para la representación del conocimiento de la ontología propuesta en la investigación.

Lenguaje de Consulta a ontologías

Los lenguajes de consulta a ontologías son un conjunto de órdenes, operadores y estructuras que, organizadas permiten solicitar información de la ontología. Para la selección de uno de estos lenguajes, se tuvo en cuenta el nivel de documentación existente, el tipo de ontología que permite consultar, la potencia para construir consultas y devolver resultados y el sistema de almacenamiento que lo soporta. Se

estudiaron los lenguaje de consulta: RQL (RDF Query Language), DQL (Daml Query Language), y SPARQL (Simple Protocol and RDF Query Language).

Se seleccionó **SPARQL** como lenguaje de consulta porque posee gran cantidad de documentación publicada acerca de sus funcionalidades, la cual se encuentra fundamentada y actualizada. Permite el trabajo con ontologías en lenguajes RDF y OWL, además de ser soportado por JENA. Es flexible ante la construcción de consultas y potente en la devolución de los resultados. Para el trabajo con SPARQL se tuvo en cuenta la sintaxis básica para la construcción de las consultas, la cual se encuentra en el manual disponible en: < http://skos.um.es/TR/rdf-sparql-query/#introduction >.

Metodología de desarrollo

Según (Avison & Fitzgerald, 1995), una metodología de desarrollo de software *«es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información».* A grandes rasgos, las metodologías de desarrollo se pueden agrupar en: metodologías ágiles o ligeras y metodologías pesadas o tradicionales.

Para la selección de uno de estos enfoques se identificaron una serie indicadores, siendo estos: las capacidades y habilidades de desarrollo del equipo de trabajo, la comunicación entre los miembros del equipo, el trabajo en equipo, el compromiso del cliente, la disposición del equipo ante los cambios durante el proyecto, el tiempo de desarrollo, el tamaño del equipo y los recursos materiales disponibles.

Teniendo en cuenta estos indicadores, se concluye que el equipo de trabajo seguirá una metodología ágil, debido a que responde a necesidades y condiciones reales del equipo de desarrollo, entre las que se encuentran: amplias habilidades y capacidades en el desarrollo de soluciones de software, dominan un lenguaje común, poseen amplias habilidades y condiciones de colaboración lo que contribuye a la confianza y al respeto mutuo, cuentan además con habilidades para la toma de decisiones y son identificados por tener una organización propia, lo cual demuestra una dinámica de funcionamiento rápida, eficiente y eficaz. Además, el cliente está altamente comprometido y motivado con el desarrollo del subsistema y el equipo de trabajo muestra gran disposición y capacidad ante los nuevos cambios surgidos durante el proyecto. A su vez, el equipo de desarrollo es pequeño y el tiempo que se dispone para el desarrollo es limitado.

Dentro de las metodologías ágiles se analizaron XP (eXtreme Programming), Scrum y OpenUP (Open Unified Process), por su gran aceptación en el desarrollo de los proyecto de software. Para ello se tomaron en cuenta una serie de indicadores: participación del cliente, realización de Pruebas, nivel de documentación que generan y nivel de flexibilidad para el equipo de trabajo.

Ante estos indicadores el equipo de desarrollo seleccionó la metodología **OpenUP** por las ventajas que ofrece esta con respecto a las demás, ejemplo de esto se evidencia en la documentación que esta genera,

la cual no es abundante pero si suficiente para la continuación del producto teniendo en cuenta que es la esencial. Además, esta metodología brinda la posibilidad al equipo de desarrollo de organizar su trabajo y determinar la mejor forma de alcanzar los objetivos, lo que motiva a los miembros del equipo a realizar un mejor trabajo. Dicha metodología permite realizar varias pruebas en cada iteración, lo cual garantiza que se cree una solución estable y disponible a medida que progresa en el desarrollo. Por otra parte la participación del cliente a diferencia de las restantes metodologías analizadas, consiste en que el mismo puede ser consultado tantas veces como sea necesario aunque no esté disponible a tiempo completo para el equipo de desarrollo.

Lenguaje de Modelado

Al realizar un software debe establecerse una forma estándar de comunicación entre todos los involucrados al software. Los lenguajes de modelado permiten representar un software y describir las actividades a realizar para su desarrollo. En la investigación se utiliza para estandarizar la documentación el Lenguaje Unificado de Modelado (**UML**) y Diagrama de Flujo de Datos (**DFD**). Para la selección del lenguaje UML se tuvieron en cuenta varios indicadores como fueron, experiencia del equipo de trabajo, propósito, adaptación, herramientas de Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés de Computer Aided Software Engineering) que lo utilizan y posibilidades de extensión del lenguaje.

UML es un lenguaje de propósito general, se adapta a situaciones o necesidades específicas, permite a los usuarios extender o incluso modificar el lenguaje, además es utilizado por varias herramientas CASE como son: Visual Paradigm, Rational Rose y Enterprise Architect. UML posee gran documentación fundamentada y actualizada, además, el equipo de desarrollo posee amplias experiencias y habilidades en el trabajo con este lenguaje.

En el caso de los DFD se tuvo en cuenta porque para resolver el problema identificado lo importante no es representar la visión del usuario, sino representar la transformación del dato desde la entrada hasta la salida en el subsistema. Esta selección está dada porque los diagramas de actividades o los de máquina de estado de UML, que son los que responden al método orientado al flujo, dependen de la unidad básica de UML (caso de uso), por tanto de manera indirecta aunque son diagramas orientados al flujo, son diagramas que responden indirectamente al interés y visión del usuario. Esta notación permite representar un flujo sin estar asociado a un caso de uso, lo cual justifica la selección de esta notación y no los diagramas orientados al flujo perteneciente a UML.

Herramienta para el modelado de diagramas

La toma de una aceptada decisión a la hora de escoger una herramienta de modelado con UML, puede ser un factor importante para lograr la calidad de un proyecto. Es por ello que el equipo de desarrollo definió una serie de indicadores para la selección de la herramienta, siendo estos: la plataforma y tipo de

software, modelado de datos y manejo de diagramas, apoyo metodológico y soporte completo del UML. Para ello se estudiaron las siguientes herramientas: Visual Paradigm, Rational Rose y Enterprise Architect.

Teniendo en cuenta estos indicadores y otros como los conocimientos, experiencias y habilidades del propio equipo de desarrollo se decidió utilizar **Visual Paradigm** como herramienta CASE, para el modelado. Otro factor que implicó la selección de la misma fue el tiempo limitado para el estudio de otra herramienta.

Una vez definido el lenguaje de modelado y la herramienta CASE, se debe hacer un estudio del lenguaje de programación a utilizar en el desarrollo del subsistema.

Lenguaje de Programación

Un lenguaje de programación es un lenguaje diseñado para describir un conjunto de acciones consecutivas, permitiendo crear programas mediante instrucciones, operadores y reglas de sintaxis para comunicarse con los dispositivos hardware y software de una máquina.

Basado en el tiempo limitado que se tiene para el desarrollo del subsistema es necesario definir un lenguaje de programación que permita avanzar rápidamente. Para ello se tuvieron en cuenta varios indicadores: conocimiento y experiencia del equipo de desarrollo, integración y seguridad. Los lenguajes analizados fueron Java, PHP, C++ y Python.

Se determinó utilizar **Java** para la implementación del subsistema teniendo en cuenta que es un lenguaje orientado a objetos de gran potencialidad, robustez, seguro y multiplataforma. Además, el equipo de desarrollo tiene avanzados conocimientos sobre este lenguaje y poseen amplias experiencias, capacidades y habilidades con el desarrollo de aplicaciones en Java. Otras de las razones que determinó utilizar este lenguaje es la de lograr una satisfactoria integración con los subsistema que conforman la concepción de SASPED.

Luego de haber determinado el lenguaje de programación a emplear, se hace necesario un Entorno de Desarrollo para la implementación del subsistema.

Entorno Integrado de Desarrollo (IDE por sus siglas en inglés)

Para la selección del IDE, se analizaron una serie de aspectos, siendo estos: multiplataforma, de código abierto y que se comercialicen bajo una licencia de software libre, además se tuvieron en cuenta otros indicadores como experiencia y las habilidades y capacidades del equipo de desarrollo en esta herramienta. Se analizaron los IDE NetBeans y Eclipse por ser muy utilizado en el desarrollo de aplicaciones java.

El IDE seleccionado fue **Eclipse** por ser multiplataforma, de código abierto y que se comercializa bajo una licencia de software libre, cumpliendo esto con las políticas que se rige la Universidad, además el equipo de trabajo posee amplios conocimientos, experiencias y habilidades y capacidades en esta herramienta.

Otro factor fundamental para la decisión es el tiempo limitado que se dispone para el desarrollo del sistema lo que dificulta la posibilidad de seleccionar un IDE al cual se desconoce por completo.

Conclusiones del capítulo

- Para el proceso de análisis de datos se valoraron un conjunto de herramientas de análisis semántico de texto. Estas herramientas poseen limitaciones para darle solución al problema de la presente investigación, al carecer de un dominio específico para la Superación Pedagógica, por estar dirigidas al análisis de textos en idioma inglés, por no utilizar ontologías como base de conocimiento y no extraer información implícita del texto.
- Existen diversas formas de realizar análisis semántico a un texto. La Minería de Texto constituye el área de conocimiento a utilizar para resolver el problema identificado en la presente investigación, ya que permite extraer información explícita e implícita de un texto, y obtener así nuevo conocimiento que previamente era desconocido.
- En la Minería de Texto se pueden utilizar diferentes bases de conocimiento. Las ontologías son la forma técnica de mayor utilización en la comunidad científica para este tipo de tarea computacional enfocada en un dominio. Su utilización permite obtener mejores resultados que aquellas que se obtienen a través del uso de otras técnicas.
- Las características del subsistema a construir en la investigación, determinaron el uso de las siguientes herramientas y tecnologías: Jena como sistema de almacenamiento de ontología; SPARQL como lenguaje de consulta a ontología; Java como lenguaje de programación y eclipse como entorno de desarrollo. Además; UML y DFD como Lenguaje de Modelado y Visual Paradigm como herramienta CASE.

CAPÍTULO 2: SUBSISTEMA DE ANÁLISIS SEMÁNTICO DE TEXTO

En el presente capítulo se describe la operacionalización de la variable definida en la investigación, donde se explican las dimensiones, indicadores y escala de valoración. Además, se muestran los resultados de la caracterización del proceso de análisis de datos de la concepción de SASPED, a través de los métodos empíricos utilizados y el Modelo de Dominio para poder comprender mejor el contexto en el cual se desarrolla el subsistema. Por otra parte se especifican los requisitos funcionales y no funcionales, y se construye el Diagrama de Flujo del subsistema propuesto. Se concluye el capítulo con la definición de la arquitectura y el diseño del subsistema, así como los patrones de diseño utilizados en la concepción de la solución.

2.1. Definición conceptual y operacional de la variable

Según lo que establecen (Hernández, Fernández-Collado, & Baptista, 2006) y (Hernández & Coello, 2011) en el proceso investigativo se tuvo en consideración la definición conceptual y operacional de la variable, constituida esta por la palabra **información**, definido su concepto en la introducción de esta memoria. La determinación de las dimensiones de la variable se realizó a partir de la definición de (Onieva, 1990), la cual plantea que *«la información explícita, son ideas que el autor comunica de una forma directa y clara en un texto escrito y la información implícita: ideas que el autor no expresa de forma directa, sino sugerida».* Por otra parte, la determinación de los indicadores se realizó a partir de los resultados de los autores (Dávila, y otros, 2012). La operacionalización se presenta en la siguiente tabla.

Tabla 2: Operacionalización de la variable información

Variable Dependiente	Dimensión	Indicadores	Valores
Información	Explícita	Precisión	Alta: 70 a 100 % Media: 40 a 60 % Baja: 10 a 30 % Nulo: Entre el 0 y 0.99 no se refleja exactitud en las respuestas.
		Cobertura	Alta: 70 a 100 % Media: 40 a 60 % Baja: 10 a 30 % Nulo: Entre el 0 y 0.99 no se refleja exactitud en las respuestas.
	Implícita	Precisión	Alta: 75 a 100% Media: 45 a 70 % Baja: 15 a 25 % Nulo: Entre el 0 y 14 no se refleja exactitud en las respuestas.
		Cobertura	Alta: 75 a 100% Media: 45 a 70 % Baja: 15 a 25 % Nulo: Entre el 0 y 14 no se refleja exactitud en las respuestas

2.2. Caracterización del proceso de análisis de datos de la concepción de SASPED

En el curso escolar 2012-2013 se elaboró en la Universidad de las Ciencias Informáticas la Estrategia de Superación Pedagógica del Claustro, luego de detectar una serie de insuficiencias en la preparación de los docentes.

Para la confección de las estrategias individuales de Superación Pedagógica de cada docente es necesario determinar sus necesidades propias de superación, independientemente de su categoría docente o título académico. Estas necesidades según la estrategia se identificarán mediante un único instrumento de diagnóstico, el cual será una encuesta, realizada la misma por el Centro de Innovación y Calidad de la Educación (CICE).

Para el desarrollo de la Estrategia se ha concebido un Sistema Automatizado de Superación Pedagógica (SASPED), con el objetivo de, según (Ciudad, y otros, 2013, pág. 6), «*perfeccionar el proceso de superación pedagógica del claustro en la UCI e integrarlo con el resto de las formas existentes actualmente de superación profesional y formación académica en las Ciencias Pedagógicas y de la Educación, para contribuir a la elevación de la cultura profesional pedagógica de los actores que intervienen en la formación del Ingeniero en Ciencias Informáticas*».

En la concepción de SASPED se obtiene mediante el sistema automatizado de encuestas de la UCI, las respuestas de las preguntas cerradas y abiertas del instrumento de diagnóstico. La concepción del sistema realiza un análisis de las preguntas cerradas mediante métodos estadísticos, lo cual permite realizar un análisis cuantitativo de los datos. Mediante las preguntas cerradas el encuestado es forzado a elegir entre alternativas que quizás no se ajusten a la respuesta deseada o a expresar una actitud aún no formada, además de limitarse a respuestas muy cortas y en ocasiones ninguna de estas respuestas describen con exactitud la opinión del encuestado, por lo que no siempre se captura la perspectiva del investigado.

Por otra parte, a las preguntas abiertas no se le realiza ningún tipo de análisis que posibilite extraer información del texto. Esto tiene como consecuencia que el investigador no cuente con información explícita e implícita, omitiendo la obtención de conocimiento potencialmente útil para el análisis y la toma de decisiones.

Sin embargo, realizar estos dos tipos de análisis posibilita una integración y discusión conjunta para realizar inferencias producto de toda la información recopilada, y así lograr un mayor entendimiento del fenómeno en estudio. Esto a su vez permite tener una perspectiva más amplia, ya que se obtiene información íntegra, lo cual posibilita éxito al presentar los resultados.

La caracterización realizada anteriormente evidencia insuficiencias en el proceso de análisis de datos de la concepción de SASPED, que hacen que este no contribuya a la extracción de información explícita e implícita en las respuestas a las preguntas abiertas, para ser utilizadas en la confección de las estrategias

individuales de Superación Pedagógica. Por lo anterior se constata la existencia del problema científico y se corrobora el objetivo propuesto.

2.3. Modelo de dominio

Con el objetivo de describir y expresar el contexto en que se desarrolla el subsistema y en aras de puntualizar la situación descrita anteriormente, se elaboró a nivel de proyecto un modelo de dominio general, el cual integra la concepción de SASPED (ver Anexo 1); de este modelo los conceptos que son pertinentes para la presente investigación se encuentran en la siguiente figura.

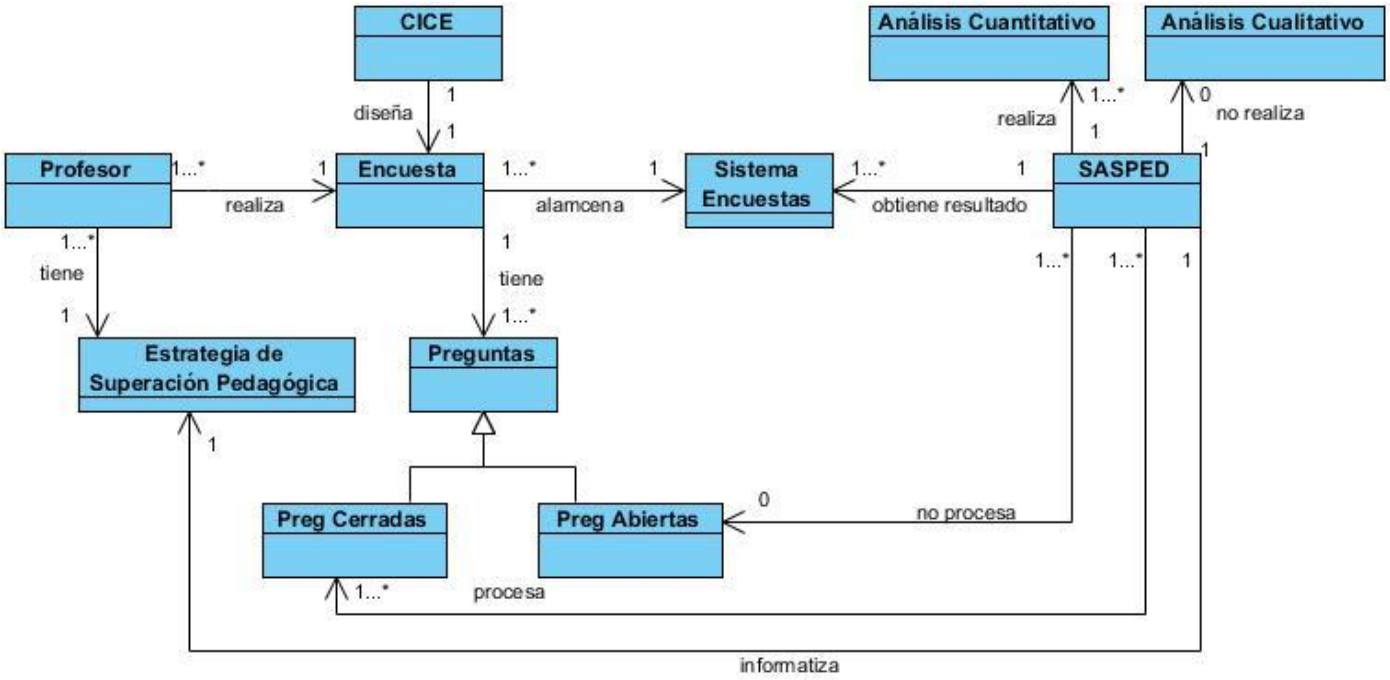


Figura 4: Modelo de dominio del subsistema

Para una mejor comprensión del Modelo de Dominio del subsistema, se proporciona un marco conceptual con las definiciones identificadas en el contexto del proceso de análisis de datos de la concepción de SASPED, estas son:

Profesor: persona que realiza la encuesta para determinar sus necesidades de superación pedagógica.

CICE: Centro de Innovación y Calidad de la Educación encargado de la elaboración de la encuesta.

Encuesta: instrumento de diagnóstico para determinar las necesidades de superación pedagógica.

Preguntas Cerradas: tipo de pregunta en las que el encuestado debe limitarse en su respuesta a una de las alternativas provistas, sin posibilidad de generar mayor información.

Preguntas Abiertas: tipo de pregunta en las que el encuestado tiene la posibilidad de generar mayor información.

Sistema Encuestas: sistema automatizado que gestiona las encuestas aplicadas en la UCI.

SASPED: concepción del Sistema de Superación Pedagógica, el cual realiza un análisis cuantitativo de los resultados obtenidos de las preguntas cerradas y automatiza la Estrategia de Superación Pedagógica.

Análisis Cuantitativo: tipo de análisis que se realiza al procesar los datos mediante métodos estadísticos.

Análisis Cualitativo: tipo de análisis que se realiza al procesar los datos mediante el análisis semántico.

Estrategia de Superación Pedagógica: es una guía a seguir por los profesores para elevar su cultura profesional pedagógica. La misma se encuentra en correspondencia con las necesidades de superación detectadas en la encuesta realizada.

Propuesta del subsistema

Tomando como base los referentes teóricos-metodológicos abordados en el primer capítulo de la presente investigación, y la visión general del Modelo de Dominio descrito anteriormente, se presenta la propuesta del subsistema de análisis semántico de texto. El subsistema permite realizar un análisis semántico a las respuestas de las preguntas abiertas en la encuesta realizada a los docentes, con el objetivo de extraer información explícita e implícita de estas respuestas, y así obtener conocimiento desconocido, el cual facilitaría la toma de decisiones a la hora de trazarle la estrategia individual a cada docente. Para esto, el subsistema implementa un algoritmo de extracción de información basado en una ontología, definido el mismo en el Capítulo 1 de la memoria.

2.4. Especificación de Requerimientos de Software

(Sommerville, 2005) considera que *«los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. El proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se denomina ingeniería de requisitos»*. Según (Pressman, 2005) este proceso *«proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación, y administrar los requisitos conforme éstos se transforman en un sistema operacional»*.

A continuación se muestra un listado de los requerimientos funcionales y no funcionales del subsistema.

Requerimientos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el subsistema debe cumplir, indican qué es lo que el software debe hacer, especifican cómo debe comportarse el subsistema en situaciones particulares y cómo debe ser el comportamiento de entrada y salida del subsistema.

El levantamiento de requerimientos para el subsistema propuesto arrojó como requerimientos funcionales los siguientes:

RF 1: Obtener texto de entrada proporcionado por el sistema.

RF 2: Segmentar el texto de entrada en oraciones.

RF 3: Extraer las palabras (tokens) de cada una de las oraciones.

RF 4: Realizar un análisis morfo-sintáctico de cada uno de los tokens.

RF 5: Extraer las frases conceptuales según la categoría gramatical.

RF 6: Extraer los conceptos de la base de conocimiento asociados a las palabras claves.

RF 7: Extraer las relaciones explícitas e implícitas de los conceptos obtenidos.

RF 8: Devolver el texto procesado al sistema.

Requerimientos No Funcionales

(Jacobson, Booch, & Rumbaugh, 2000, pág. 110) son del criterio que *«los requerimientos no funcionales especifican propiedades o cualidades que el producto debe tener, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad»*. El levantamiento de requerimientos para el subsistema propuesto arrojó como requerimientos no funcionales los siguientes:

Eficiencia	
RnF 1.	El subsistema respetará las buenas prácticas de programación para incrementar el rendimiento en operaciones costosas. Se optimiza el trabajo con cadenas y otras buenas prácticas que ayudan a mejorar el rendimiento.
Usabilidad	
RnF 2.	Todo el proceso debe ser transparente para el usuario.
Confiabilidad	
RnF 3.	Se debe garantizar un tratamiento adecuado de las excepciones y validación de las entradas.
Portabilidad	
RnF 4.	El subsistema estará desarrollado de forma tal que funcione en los sistemas operativos GNU/Linux o Windows y garantizar así un producto multiplataforma.
Restricciones en el diseño y la implementación	
RnF 5.	Para la modelación del subsistema se utilizará el lenguaje UML haciendo uso de la herramienta Visual Paradigm. Se requiere el uso de la arquitectura flujo de datos y patrón tubería y filtro implementado en el lenguaje de programación Java.
Software	
RnF 6.	Se debe utilizar como base de conocimiento una Ontología de Dominio en lenguaje OWL. La utilizada está enmarcada en el dominio de Superación Pedagógica aunque el algoritmo es compatible con otras Ontologías de Dominio.
RnF 7.	Debe estar instalada la herramienta TreeTagger y se debe utilizar Jena como Framework para la

	gestión de la Ontología.
Hardware	
RnF 8.	Proporcionar características mínimas de hardware a las estaciones de trabajo. • Procesador: Pentium 4 (1.70 GHZ). • Memoria RAM: 1 GB.

2.5. Propuesta de solución

Con el objetivo de describir lo que requiere el cliente y establecer una base para la creación de un diseño de software, se puede recurrir a los siguientes métodos de modelado: basado en escenarios, basado en clases, orientado al flujo y de comportamiento; los cuales permiten al equipo de desarrollo representar el software desde diferentes perspectivas, para así aumentar la comunicación con el cliente y entre los miembros del equipo. Estos métodos de modelado no se utilizan unos u otros, indistintamente de forma exclusiva, sino que el equipo de desarrollo hace una selección de cuáles son los métodos que permiten tener una visión significativa de los requisitos del software. Para autores como (Pressman, 2005), *«el cuestionario no es cuál es el mejor, sino qué combinación de representaciones le proporcionará a los interesados el mejor modelo de requisitos de software y el puente más efectivo para el diseño de software».*

Teniendo en cuenta los elementos planteados anteriormente, en la presente investigación se utiliza como método de modelado el de **Elementos orientados al flujo,** dado que proporciona un conocimiento adicional de los requisitos y del flujo del sistema, pues se describen las transformaciones que sufre la información de entrada al sistema y la salida que genera el procesamiento. Para ello, se utiliza el diagrama de flujo, que muestra la manera en que una entrada se transforma en una salida conforme los objetos de datos se mueven a través del sistema.

Para clarificar los requisitos identificados, los cuales responden al algoritmo propuesto de análisis semántico de texto, se realiza el diagrama de flujo presentado en la figura cinco, compuesto por seis operaciones (Segmentar Texto, Extraer Tokens, Realizar Análisis Morfo-Sintáctico, Extraer Conceptos, Extraer Relaciones y Generar Resultado).

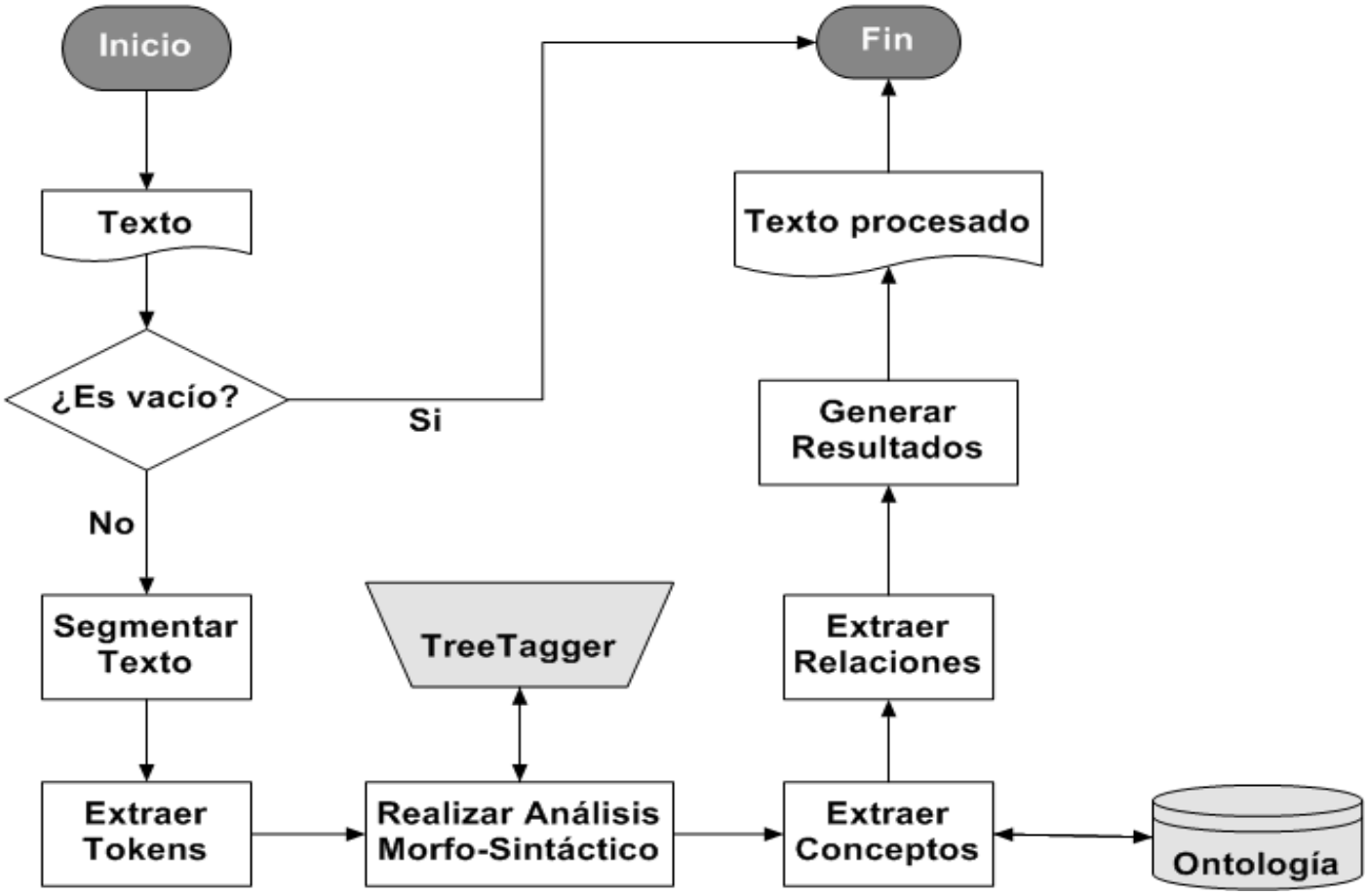


Figura 5: Diagrama de flujo del algoritmo propuesto

A continuación se explican cada una de las operaciones del algoritmo, las cuales fueron fundamentadas en el **Capítulo 1,** y se muestra el diagrama de flujo correspondiente a cada operación para lograr un mayor entendimiento.

Segmentar Texto

Esta operación recibe como entrada un texto en lenguaje natural enviado por SASPED, desfragmenta el texto por oraciones teniendo en cuenta la función del punto y devuelve como resultado una lista de oraciones.

Entrada: Texto en lenguaje natural.

Procedimiento:

- Obtiene el texto por parámetro.
- Identifica cada una de las oraciones según la función del punto en el texto.
- Almacena las oraciones en una lista.

Salida: Lista de Oraciones.

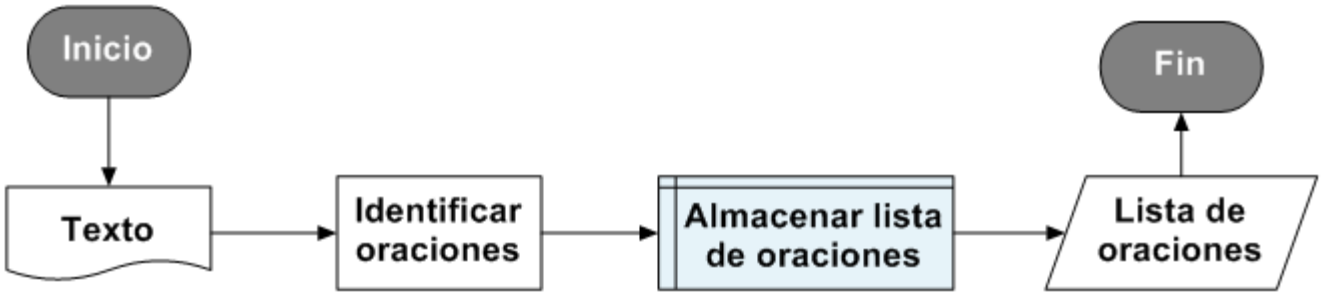


Figura 6: Diagrama de flujo de la operación Segmentar Texto

Extraer Tokens

Esta operación recibe como entrada una lista de oraciones compuesta por palabras, signos de puntuación, números, entre otros; y se encarga de identificar cada una de las partes de las oraciones (tokens), devolviendo como resultado una lista de tokens por cada oración.

Entrada: Lista de Oraciones.

Procedimiento:

- Obtiene un caracter de la lista de oraciones.
- Mientras identifica una letra se almacena el caracter.
- Cuando identifica un caracter distinto de una letra o un espacio en blanco, construye un token con los caracteres identificados anteriormente y otro con el carácter actual.
- Almacena los tokens en una lista.

Salida: Lista de Tokens.

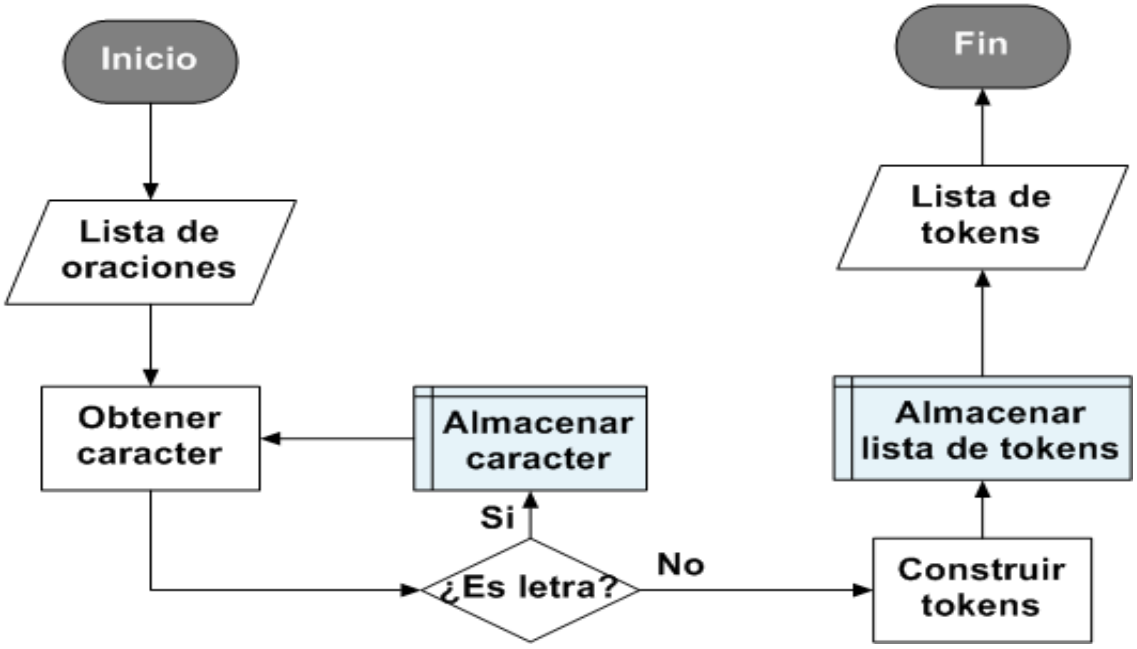


Figura 7: Diagrama de flujo de la operación Extraer Tokens

Realizar Análisis Morfo-Sintáctico

Esta operación recibe como entrada una lista de tokens identificados en la operación anterior, se determina la raíz morfológica (RM) y categoría gramatical (CG) de cada uno de los tokens y devuelve una lista de tokens con su CG y RM.

Entrada: Lista de tokens.

Procedimiento:

- Obtiene un token de la lista de entrada.
- Identifica el idioma con el cual se procesará el token.
- Identifica por cada token su CG y RM haciendo uso de la herramienta de etiquetado gramatical TreeTagger.
- Almacena los tokens con su CG y RM en una lista.

Salida: Lista de Tokens con su CG y RM.

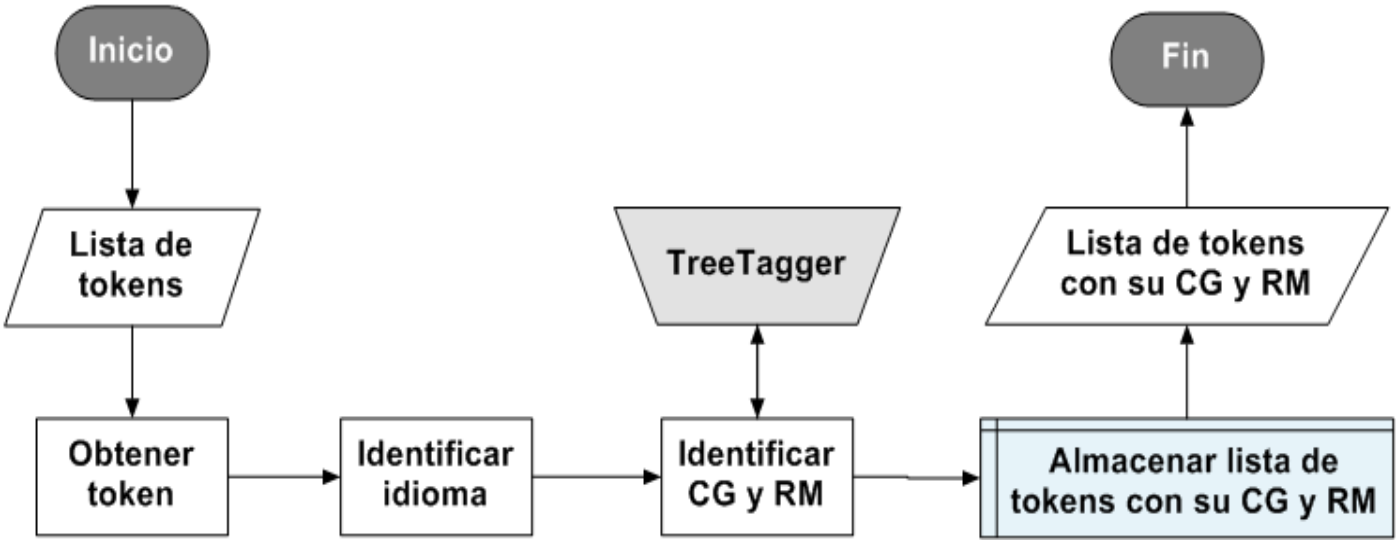


Figura 8: Diagrama de flujo de la operación Realizar Análisis Morfo-Sintáctico

Extraer Conceptos

Esta operación recibe como entrada la lista de tokens con su CG y RM, determina los conceptos asociados a la base de conocimiento (ontología) y devuelve la lista de conceptos que coinciden con la ontología.

Entrada: Lista de Tokens con su CG y RM.

Procedimiento:

- Obtiene un token de la lista de entrada.
- Determina si este token junto a su predecesor constituyen una estructura multipalabra

(lugares, nombres propios, organizaciones).

- Almacena estas estructuras multipalabras, así como los restantes sustantivos propios (NP), sustantivos comunes (NC) y adjetivos (ADJ) presentes en el texto.
- Extrae los conceptos de la ontología (clases e individuos).
- Compara los conceptos con los tokens almacenados, para determinar los conceptos asociados a la ontología.
- Almacena en una lista los conceptos asociados a la ontología**.**

Salida: Lista de conceptos asociados a la ontología.

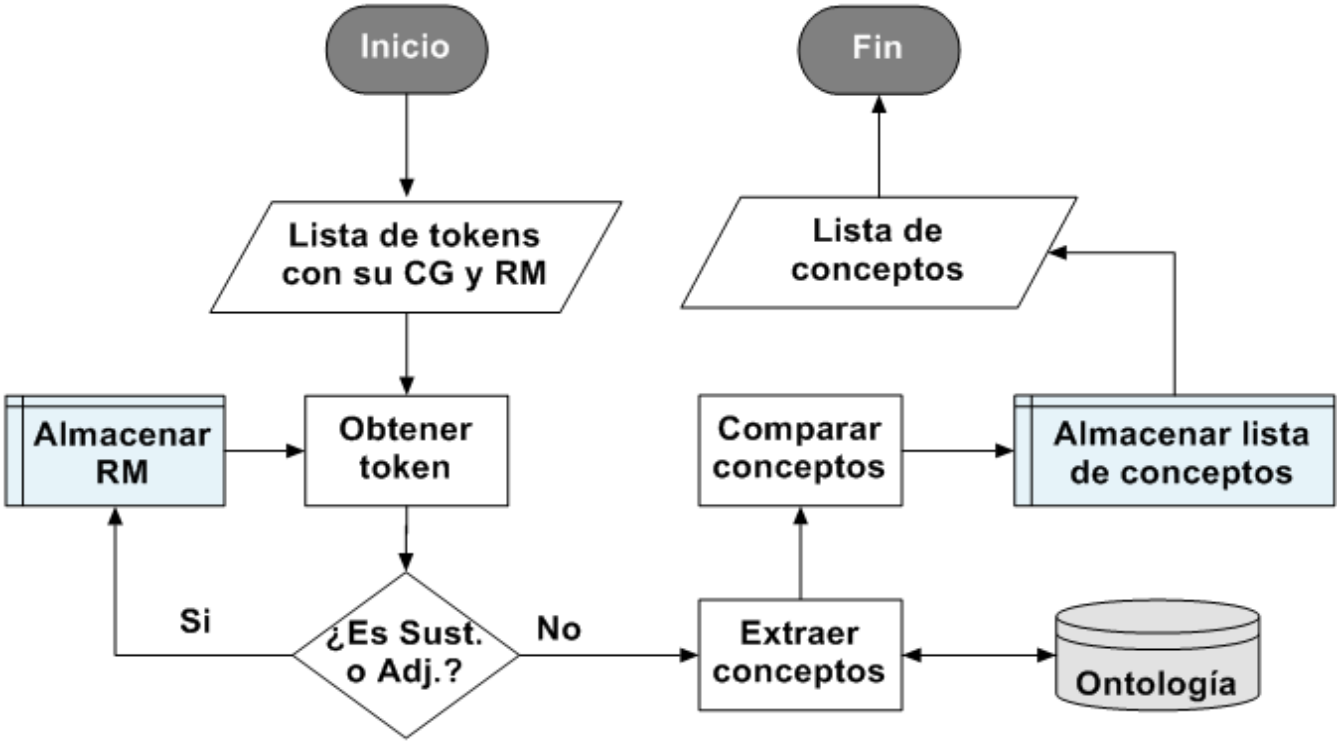


Figura 9: Diagrama de flujo de la operación Extraer Conceptos

Extraer Relaciones

Esta operación recibe como entrada una lista de conceptos asociados a la ontología, determina las relaciones explícitas e implícitas del texto almacenadas en la ontología y devuelve una lista con las relaciones de los conceptos.

Entrada: Lista de conceptos asociados a la ontología.

Procedimiento:

- Obtiene la RM asociada a un concepto de la lista de conceptos.
- Dado la RM obtiene los tipos de relaciones asociadas a él y sus atributos.

- Dado la RM construye una consulta (teniendo en cuenta los tipos de relaciones asociadas a él y sus atributos) y obtiene los valores de las relaciones asociadas a la RM.
- Almacena las relaciones asociadas a la RM en una lista.

Salida: Lista de relaciones asociadas a la ontología.

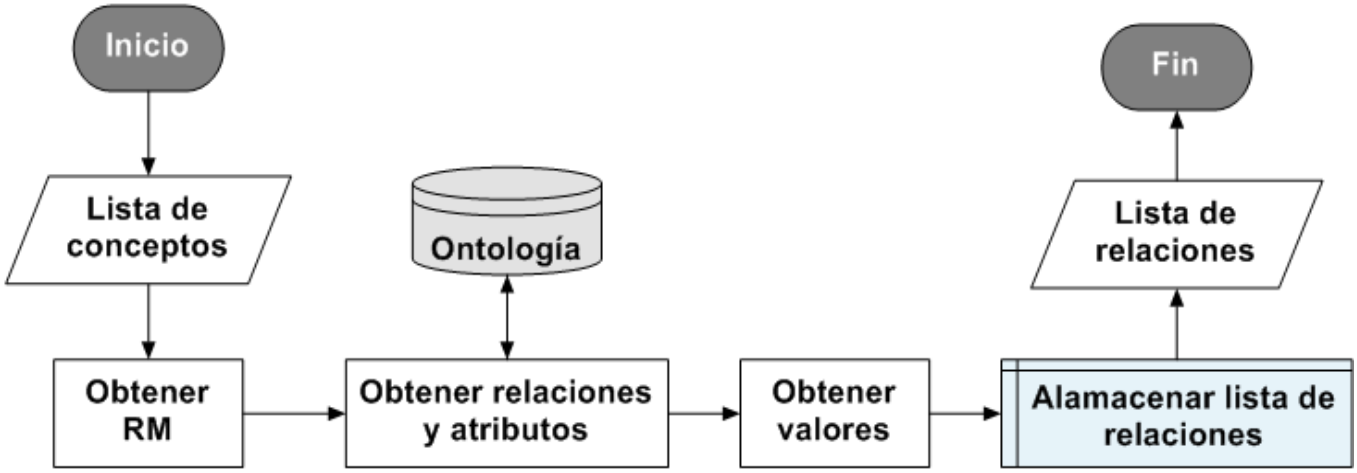


Figura 10: Diagrama de flujo de la operación Extraer Relaciones

Generar Resultado

Esta operación recibe como entrada una lista con las relaciones asociadas a la ontología y devuelve un texto como resultado final del algoritmo, el cual contiene la información explícita e implícita del texto inicial.

Entrada: Lista de relaciones asociadas a la ontología.

Procedimiento:

- Obtiene cada uno de los conceptos y sus relaciones de la lista de entrada.
- Construye el texto de salida, recorriendo la lista de entrada.
- Almacena el texto.

Salida: El texto conformado por la información explícita e implícita del texto inicial.

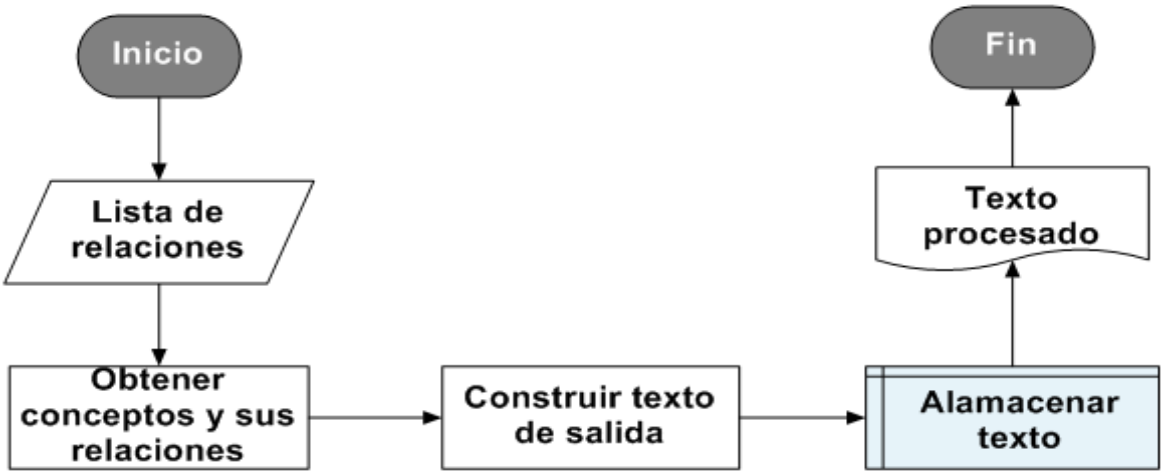


Figura 11: Diagrama de flujo de la operación Generar Resultados

2.6. Arquitectura y Diseño del subsistema

Uno de los pasos fundamentales en el proceso de desarrollo de software es la definición de la arquitectura del software, la cual constituye el diseño de más alto nivel de la organización de un sistema, programa o aplicación.

(Pressman, 2005) plantea que *«la arquitectura de software es la estructura u organización de los componentes del programa (módulos), la manera en qué estos componentes interactúan, y la estructura de datos que utilizan los componentes».*

Por otra parte las autoras (Almeira & Cavenago, 2007) concluyen que «*la arquitectura brinda una visión global del sistema. Esto permite entenderlo, organizar su desarrollo, plantear la reutilización del software y hacerlo evolucionar. Esto exige diseñar muy cuidadosamente la arquitectura bajo la cual funciona el sistema, ya que las decisiones que se tomen tendrán gran influencia a lo largo de todo el ciclo de vida de la aplicación»*. Teniendo en cuenta lo planteado por los autores citados anteriormente se define el estilo y patrón arquitectónico a seguir, conducido los mismos por las propiedades generales del software en cuestión.

Estilo y patrón arquitectónico

El estilo arquitectónico **Flujo de datos** es el empleado en la solución del subsistema propuesto, dado que en la solución del problema se necesita como entrada una fuente de datos (texto), dividir las tareas del subsistema en varios pasos donde se procesan un flujo de datos y devolver un resultado procesado. Para satisfacer esas condiciones el estilo y patrón arquitectónico de mayor probabilidad de ofrecer resultados positivos es Flujo de Datos con el patrón Tuberías y Filtros.

Esta conclusión se basa en que el subsistema se divide en varios pasos de procesamiento, donde ocurren una serie de transformaciones a partir de la fuente de datos. Esta fuente de datos representa la entrada al subsistema, proporcionada la misma por SASPED, la cual fluye a través de una serie de componentes (filtros) conformados por: Segmentar texto, Extraer tokens, Realizar Análisis Morfo-Sintáctico, Extraer

conceptos y Extraer relaciones. Estos filtros se conectan a través de tuberías por donde transita el flujo de datos, transformando gradualmente las entradas en salidas. Se sigue en la arquitectura un estilo **secuencial por lotes**, donde los datos de salida de un paso son la entrada para el paso siguiente.

En el siguiente diagrama se muestra una visión general de la arquitectura del subsistema, donde se expresa la secuencia de los componentes, los cuales representan un estilo de flujo de datos con la utilización del patrón Tuberías y Filtros.

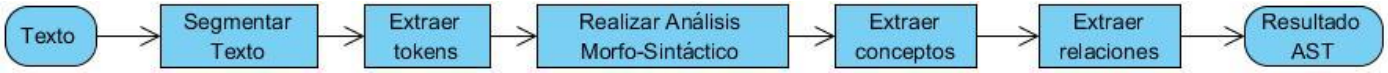


Figura 12: Arquitectura del subsistema

Modelo de Diseño

(Jacobson, Booch, & Rumbaugh, 2000, pág. 208) consideran que *«un modelo de diseño es un modelo de objetos que describe la realización física de los casos de usos centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tiene impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación»*.

Diagrama de Clases del Diseño

(Alcocer & Ortiz, 2013) plantean que *«los diagramas de clases exponen un conjunto de interfaces, colaboraciones y sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema. Son importantes para visualizar, especificar, documentar modelos estructurales y construir sistemas ejecutables aplicando ingeniería directa e inversa».* Según (Larman, 1999), *«el diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces (las de Java, por ejemplo) en una aplicación».*

En la figura 13 se muestra el diagrama de clases del diseño del subsistema, acompañado de la descripción de cada una de las clases que conforman el diagrama.

Descripción de las clases

Tabla 3: Descripción de la clase EntradaSalida

Nombre: EntradaSalida	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	obtenerTexto(texto: string)
Descripción:	El subsistema obtiene el texto enviado por el sistema, de no ser vacío crea una instancia de la CC_AST e invoca al método procesamiento de la instancia creada y envía como parámetro el texto.
Nombre:	devolverResultado(resultado: string)
Descripción:	El subsistema devuelve el resultado obtenido del análisis semántico del texto, es decir, el resultado obtenido de la llamada al método procesamiento.

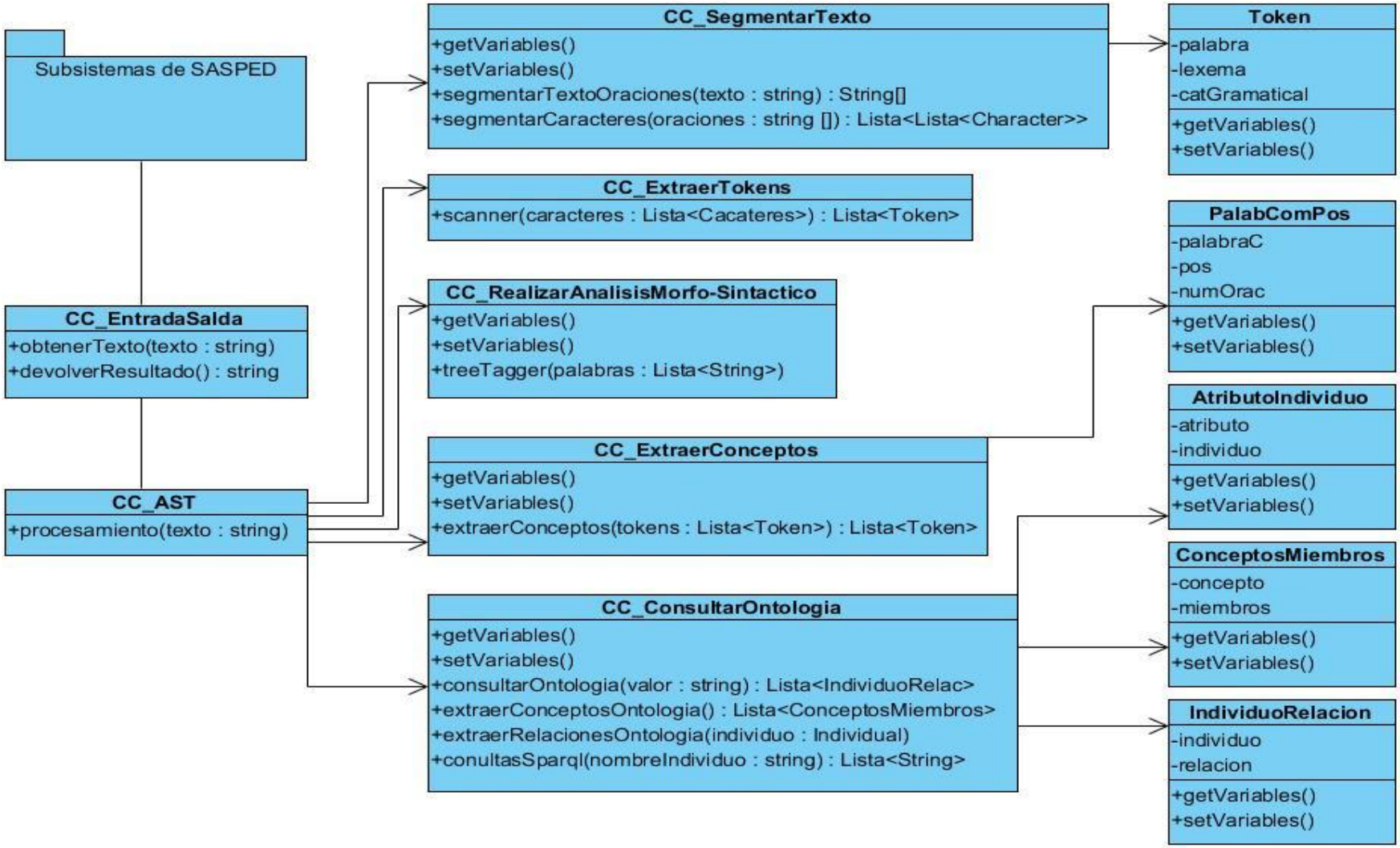


Figura 13: Diagrama de clases del diseño del subsistema

Tabla 4: Descripción de la clase AST

Nombre: AST	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	procesamiento(texto: string)
Descripción:	Método que ejecuta el algoritmo propuesto. Obtiene el texto enviado por el método obtenerTexto y crea una instancia de la CC_SegmentarTexto y de la CC_ExtraerTokens. Invoca al método segmentarTextoOraciones y al método segmentarCaracteres de la CC_SegmentarTexto. Crea los tokens con la ejecución del método scanner de la CC_ExtraerTokens e identifica su Categoría Gramatical (CG) y su Raíz Morfológica (RM) al invocar a la instancia de la CC_RealizarAnalisisMorfo-Sintactico y ejecutar el método treeTagger. Crea una instancia de la CC_ExtraerConceptos, invoca el método extraerConceptos de dicha instancia y crea una instancia de la CC_ConsultarOntologia e invoca al método extraerConceptosOntologia y obtiene solo los conceptos que coinciden con los de la ontología. Invoca al método extraerRelacionesOntologia y obtiene las relaciones de los conceptos.

Tabla 5: Descripción de la clase SegmentarTexto

Nombre: SegmentarTexto	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	segmentarTextoOraciones(texto: string)
Descripción:	Obtiene el texto enviado por el método procesamiento, verifica que no sea vacío, desfragmenta el texto por oraciones con la utilización de la función Split de java y devuelve las oraciones. De ser vacío el texto devuelve un mensaje que indica “texto vacío”.

Tabla 6: Descripción de la clase ExtraerTokens

Nombre: ExtraerTokens	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	scanner(caracteres: Lista<Caracteres>)
Descripción:	Obtiene la lista de caracteres enviados por el método procesamiento. Recorre cada uno de los caracteres y crea los tokens según los delimitadores de los mismos.

Tabla 7: Descripción de la clase RealizarAnalisisMorfo-Sintactico

Nombre: RealizarAnalisisMorfo-Sintactico	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	treeTagger()
Descripción:	Obtiene la lista de tokens enviada por el método procesamiento y utiliza la herramienta TreeTagger previamente instalada. Se especifica la dirección donde se encuentra instalada la herramienta TreeTagger y el idioma con el que se utilizará. Por cada token identifica su CG y RM.

Tabla 8: Descripción de la clase ExtraerConceptos

Nombre: ExtraerConceptos	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	extraerConceptos(tokens: Lista<Token>)
Descripción:	El método recibe la lista de tokens enviada por el método procesamiento, identifica las palabras compuestas, los sustantivos y las formas verbales y devuelve una lista con lo identificado anteriormente.

Tabla 9: Descripción de la clase ConsultarOntologia

Nombre: ConsultarOntologia	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	extraerConceptosOntologia()
Descripción:	Este método utiliza las funcionalidades del framework JENA para acceder a la ontología. Obtiene de la Ontología las clases y sus miembros que no son más que los conceptos del dominio en que se ejecuta el algoritmo.
Nombre:	extraerRelacionesOntologia(individuo Individuo)
Descripción:	Este método utiliza las funcionalidades del framework JENA para acceder a la ontología. Obtiene la lista de relaciones asociadas a un miembro dado.
Nombre:	consultasSparql(concepto: string)
Descripción:	Recibe por parámetro un concepto, utiliza las funcionalidades del framework JENA para obtener de la ontología las relaciones y los atributos del concepto. Con las relaciones y los atributos construye una consulta Sparql y devuelve la consulta creada.
Nombre:	consultarOntologia(valor: string)
Descripción:	Recibe por parámetro un concepto. Se especifica la dirección de la ontología y obtiene la lista de consultas Sparql con la llamada al método consultasSparql devolviendo las relaciones asociadas al concepto.

Patrones de Diseño

(Gamma, Helm, Johnson, & Vlissides, 2003) tienen la opinión que *«un patrón de diseño constituye una solución estándar para un problema común de programación en el desarrollo del software. Además es una técnica muy eficaz para flexibilizar el código haciéndolo satisfacer ciertos criterios, así como permite una manera más práctica de describir ciertos aspectos de la organización de un programa»*.

Para el desarrollo del subsistema se tuvo en cuenta los patrones de asignación de responsabilidades, conocidos como patrones **GRASP,** acrónimo de "General Responsibility Assignment Software Patterns", los cuales según (Larman, 1999) *«describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones».*

Además se utilizaron los patrones **Gof** acrónimo de "Gang of Four" (Banda de los Cuatros), los cuales se clasifican en dependencia del propósito para los que hayan sido definidos: de creación, estructurales y de comportamiento.

En la figura 13 se puede observar la utilización de los patrones GRASP y Gof, ejemplificándose a continuación.

Patrones GRASP

Patrón Experto en la clase **ConsultarOntologia:** esta clase contiene la información necesaria para cumplir esa responsabilidad.

Patrón Creador en la clase **SegmentarTexto:** esta clase es la responsable de crear una instancia de la clase Token dado que utiliza objetos de esta clase.

Patrón Controlador en la clase **AST:** esta clase es la encargada de atender este evento en el sistema, definiendo las principales operaciones que deben realizarse.

Patrón Alta cohesión y **Bajo acoplamiento**: se observan de forma general en el diagrama, debido a que las dependencias entre las clases es mínima, y la responsabilidad que realiza cada una están bien enfocadas y relacionadas.

Patrón Gof

Patrón Fachada/Facade en la clase **EntradaSalida:** esta clase reduce la lógica de negocio, es decir, simplifica el acceso a las clases del subsistema, la cual proporciona una única clase que se utiliza para comunicarse con el conjunto de clases. Reduce la complejidad y minimiza las dependencias.

Conclusiones del Capítulo

- La caracterización del proceso de análisis de datos de la concepción de SASPED refleja insuficiencias en el análisis de los tipos de preguntas, lo que limita obtener información explícita e implícita para ser utilizadas en la confección de la estrategia individual de Superación Pedagógica a cada docente.
- El análisis del dominio de aplicación junto a las necesidades del cliente, permitió identificar ocho requisitos funcionales y no funcionales, para garantizar el correcto funcionamiento del subsistema de análisis semántico.
- El diseño de la solución, se divide en varios pasos de procesamiento para el análisis semántico de texto, donde ocurren en cada uno de ellos una serie de transformaciones a partir de la fuente de datos hasta generar la salida de los datos procesados, lo cual es satisfecho por la arquitectura “Flujo de Datos”.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DEL SUBSISTEMA

En el presente capítulo se ejemplifica la implementación de cada una de las operaciones que componen el algoritmo propuesto y se exponen los estándares de codificación, así como el tratamiento de errores empleado en la solución. Se determina la técnica de prueba de software a utilizar, que permita comprobar la calidad de la solución obtenida. Se detallan los casos de prueba y se analizan los resultados obtenidos. Además, se evalúa la variable de la investigación con el uso del subsistema Análisis Semántico de Texto para SASPED, donde se utiliza como método científico el experimento. Se culmina el capítulo con el análisis de los resultados y la demostración de la hipótesis planteada.

3.1. Implementación

(Pressman, 2005) considera que *«la implementación es el principal flujo de trabajo en la fase de construcción. En él se describe cómo los elementos del modelo de diseño se implementan en función de componentes y por ende en piezas más manejables por el lenguaje del programador. Tiene como objetivo llevar a cabo la implementación de cada una de las clases significativas del diseño»*.

Para lograr un mejor entendimiento de la implementación del algoritmo propuesto, se ejemplifica cada una de las operaciones por la cual fluye el texto de entrada, siendo en este caso la primera respuesta presentada en el Anexo 3. Como se expuso en el capítulo 1, el algoritmo ejecuta un conjunto de operaciones una vez obtenido un texto en lenguaje natural no estructurado comenzando por la operación Segmentar Texto, la cual utiliza la función SPLIT de JAVA para desfragmentar el texto en oraciones, teniendo en cuenta el punto en el mismo. A continuación se muestra como es generada la lista de oraciones al ser procesado el texto por la operación descrita anteriormente. Ver figura 14.

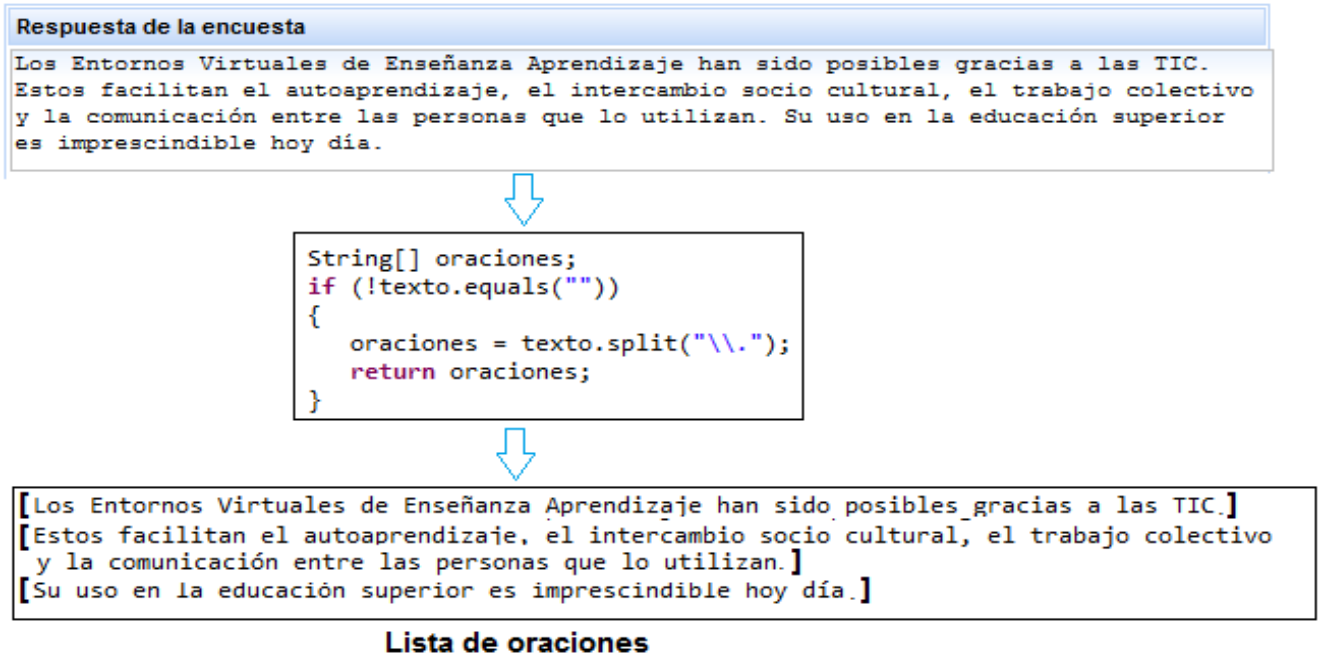


Figura 14: Procesamiento de la operación Segmentar Texto

La operación Extraer Token recibe como entrada la salida de la operación anterior y utiliza las sentencias "while" y "switch case" de java para identificar cada uno de los tokens en la oración, al recorrer los caracteres que la conforman y teniendo en cuenta sus fronteras. Ver figura 15.

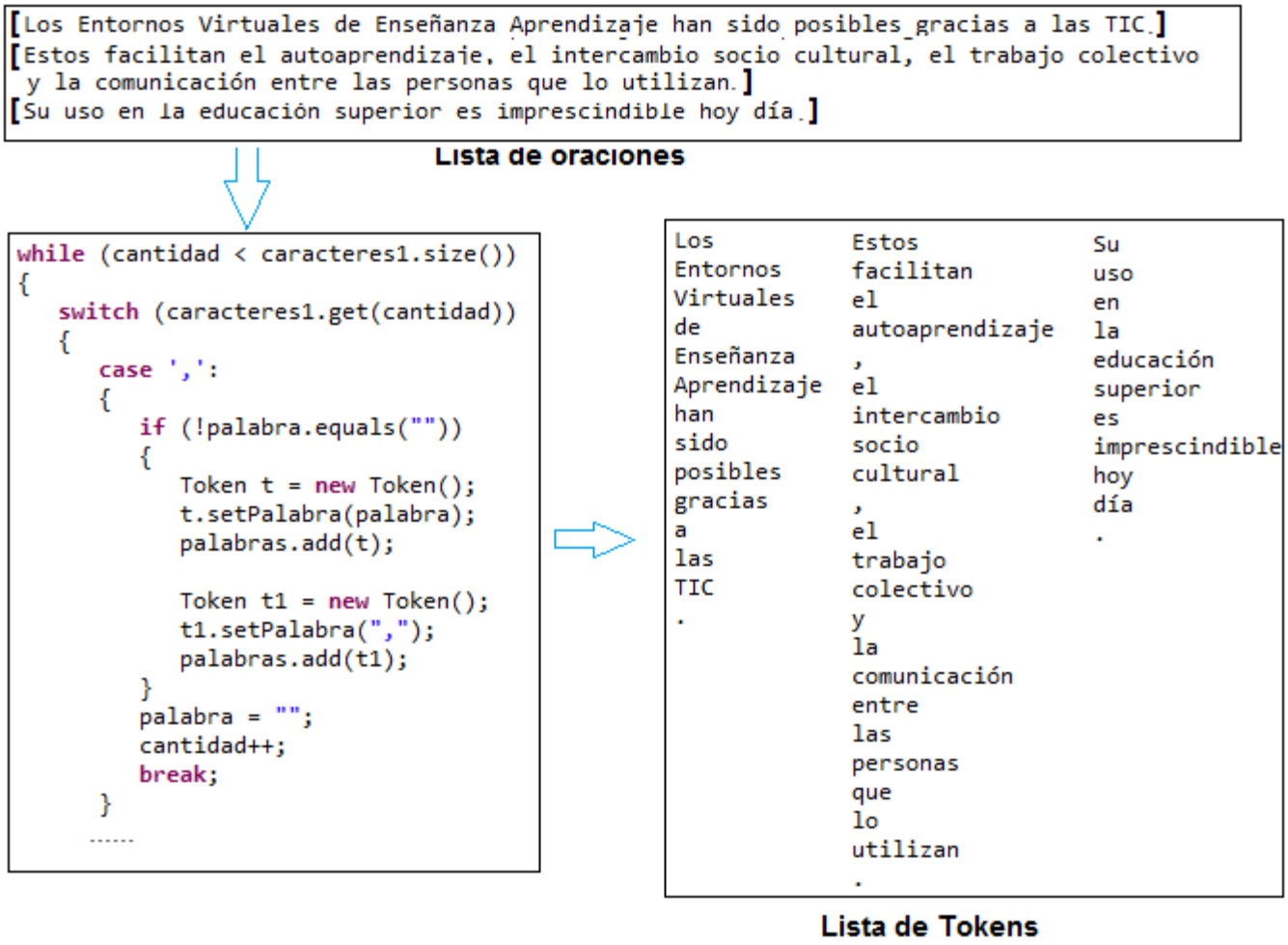


Figura 15: Procesamiento de la operación Extraer Token

La operación de Realizar Análisis Morfo-Sintáctico recibe como entrada la salida de la operación anterior y utiliza la herramienta TreeTagger para identificar la CG y RM de cada uno de los tokens, mediante el lenguaje de programación java. Para el correcto funcionamiento de esta herramienta se le indica la dirección donde se encuentra instalada y el idioma con el cual se procesará el texto. De esta forma se obtiene como resultado el token analizado, seguido por la etiqueta de acuerdo a su categoría gramatical y por último la raíz o lema del mismo. Ver figura 16.

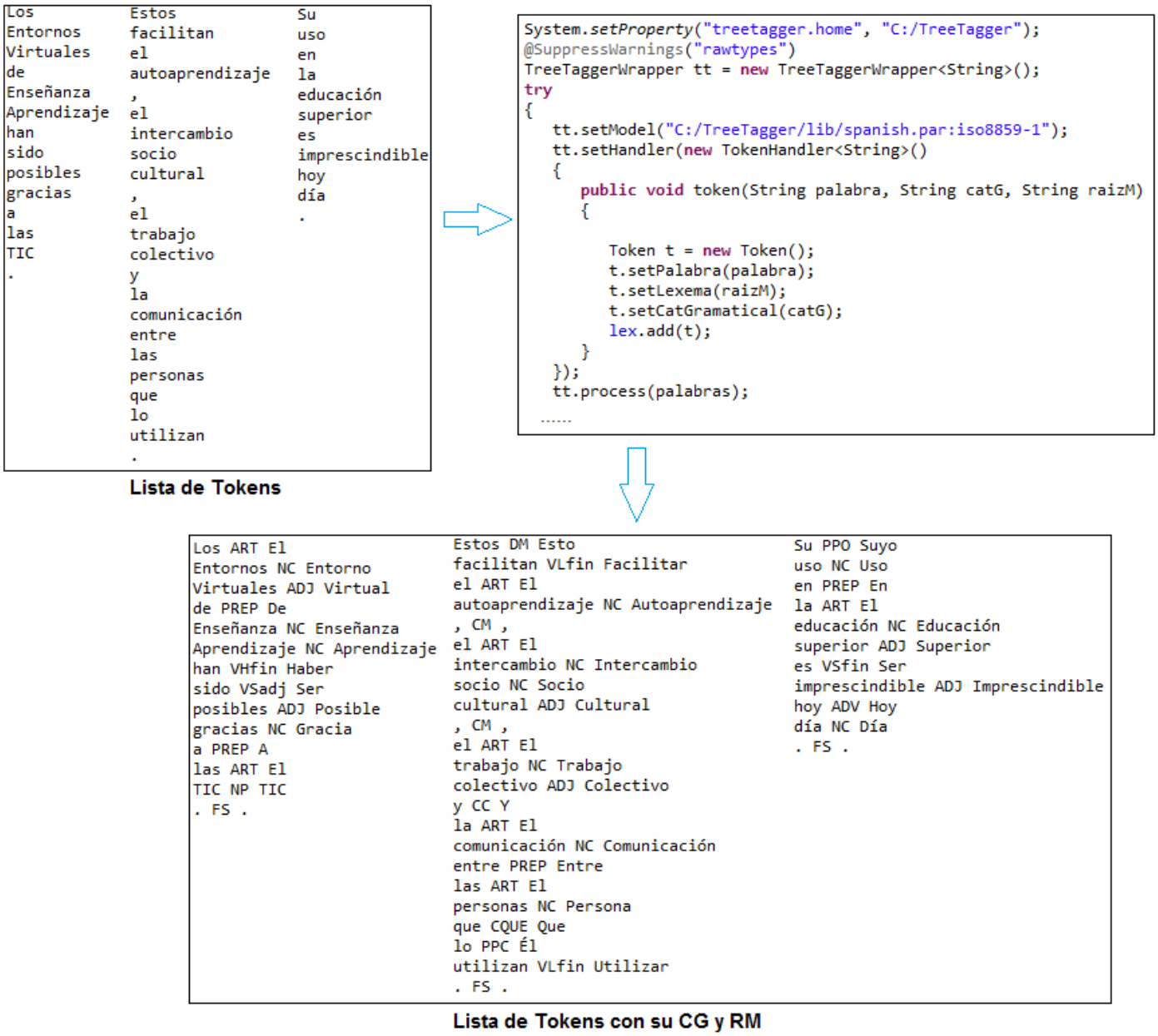


Figura 16: Procesamiento de la operación Realizar Análisis Morfo-Sintáctico

La operación de Extraer Conceptos recibe como entrada la salida de la operación anterior y utiliza el framework Jena para extraer los conceptos del texto asociados a la ontología, haciendo más fácil el trabajo con la misma. Se identifican los tokens que puedan generar conceptos teniendo en cuenta la CG y dado estos tokens devuelve aquellos conceptos que se encuentren en la ontología y estén asociados a la RM de los tokens. Ver figura 17.

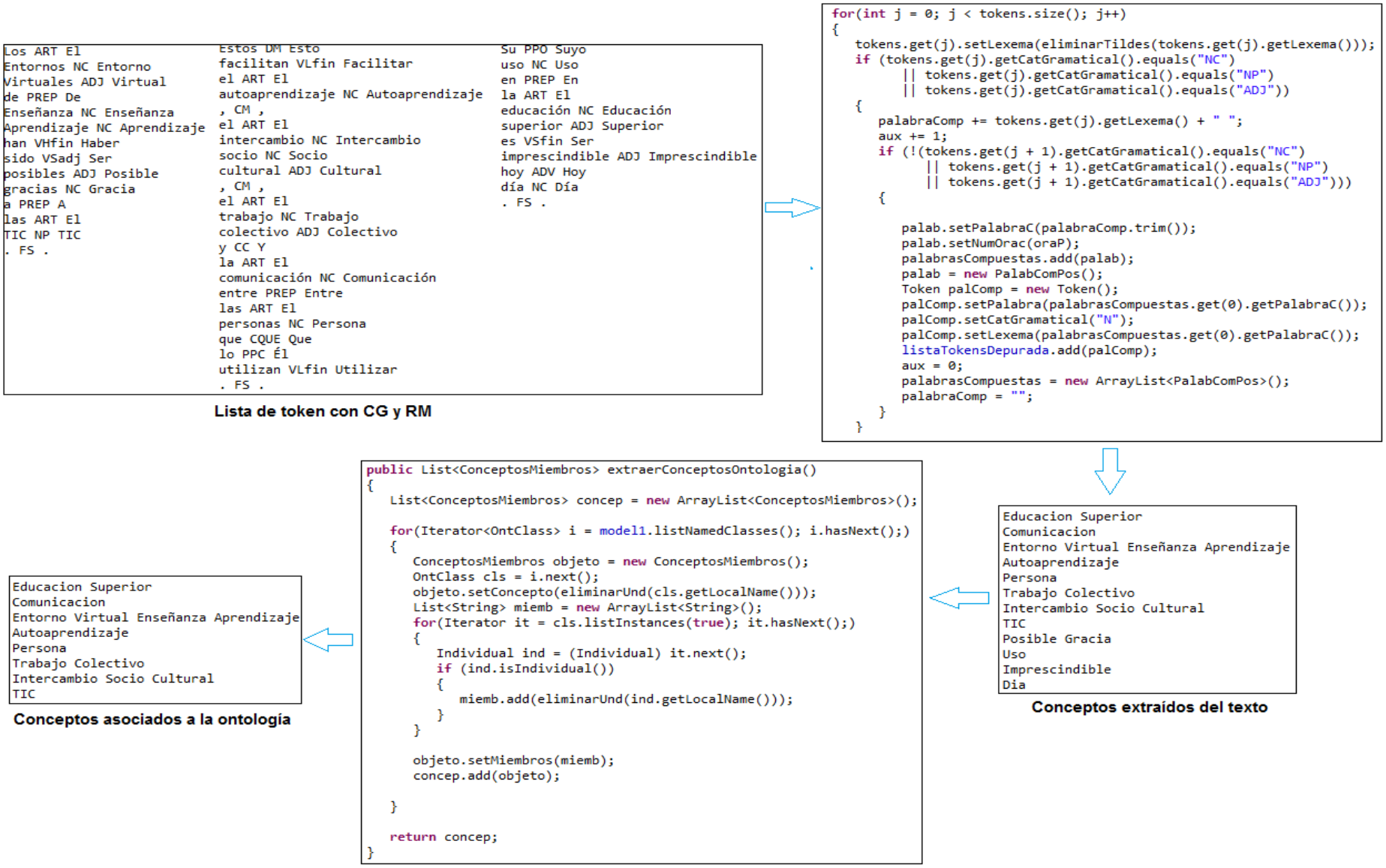


Figura 17: Procesamiento de la operación Extraer Conceptos

Por último las operaciones Extraer Relaciones y Generar Resultados utilizan el framework Jena, el lenguaje SPARQL y el trabajo con cadenas para devolver el texto de salida. Ver figura 18.

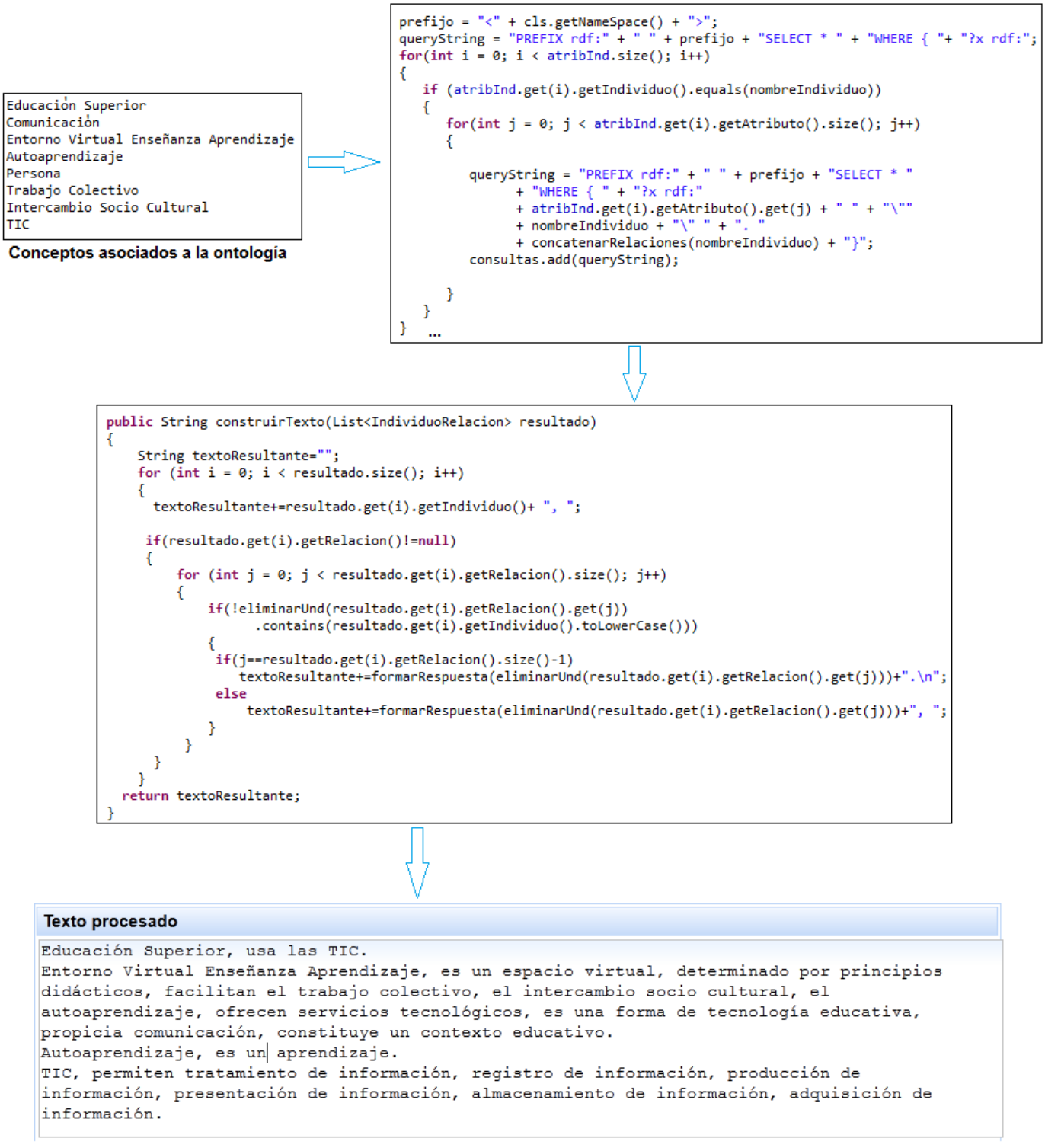


Figura 18: Procesamiento de la operación Extraer Relaciones y Generar Resultados

Tratamiento de errores

La presencia de errores durante la ejecución de un programa resulta inevitable, es por esto que un correcto tratamiento de ellos garantiza considerablemente un aumento en la calidad de las aplicaciones desarrolladas. La técnica de programación recomendada para erradicar la propagación de los errores producidos, es el tratamiento de excepciones o tratamiento de errores como también es conocido.

En el desarrollo del subsistema es utilizado el bloque **try catch** en fragmentos de código donde resulta necesario capturar y manejar determinadas excepciones. La figura 19 hace referencia a su uso en una de las clases del subsistema desarrollado.

De forma general, en el subsistema propuesto se utilizan todas las facilidades que brinda la plataforma para el tratamiento de excepciones. Para cada fragmento de código donde se espere una situación anómala, se definen las excepciones correspondientes para luego ser tratadas y evitar la interrupción del subsistema (ver tabla 10).

Tabla 10: Lista de las situaciones anómalas y respuestas del subsistema

Situación Anómala	Respuesta del Sistema
No se encuentre la dirección de la instalación del TreeTagger.	Envía el mensaje: “Error durante el acceso al TreeTagger”
No se encuentre la dirección de la librería del idioma a utilizar en el TreeTagger.	Envía el mensaje: “Error durante el acceso al idioma del TreeTagger”
No se encuentre la dirección del fichero de la ontología.	Envía el mensaje: “La ontología no está disponible”
Ocurre un error en el procesamiento del texto.	Envía el mensaje: “Ha ocurrido un error durante el procesamiento del texto ”

```
System.setProperty("treetagger.home", "C:/TreeTagger");
TreeTaggerWrapper tt = new TreeTaggerWrapper<String>();
try {
        tt.setModel("C:/TreeTagger/lib/spanish.par:iso8859-1");
        tt.setHandler(new TokenHandler<String>()
                {
                    public void token(String token, String pos, String lemma)
                    {

                        Token t = new Token();
                        t.setPalabra(token);
                        t.setLexema(lemma);
                        t.setCatGramatical(pos);
                        lex.add(t);
                    }
                }
        );
       tt.process(palabras);
 }
catch (Exception e) {
   tt.destroy();
   error="Error durante el acceso al treetagger.";
   System.out.println(error);
}
```

Figura 19: Tratamiento de errores

Estándares de codificación

Los estándares de codificación son un conjunto de reglas que guían a los desarrolladores para escribir un código fuente que sea entendible, y posea una adecuada comprensión para que en un futuro se logre dar mantenimiento al sistema. Estos estándares facilitan una mayor organización y limpieza en el código.

A continuación se describen una serie de estrategias de codificación a utilizar para la implementación del subsistema:

- Los nombres de las clases comienzan con la primera letra en mayúscula y el resto con minúscula, en caso de ser una palabra compuesta se empleará la notación PascalCasing[4].
  Ejemplo: Segmentacion.java, ExtraccionConceptos.java
- Los nombres de los métodos y los atributos de las clases comienzan con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing[5].
  Ejemplo: consultasOntologias, procesamiento.

[4] Establece que el primer carácter de todas las palabras se expresa en mayúscula y el resto de los caracteres en minúscula.
[5] Define que el primer carácter de todas las palabras, excepto la primera palabra se expresa en mayúscula y el resto de los caracteres en minúscula.

- El inicio y fin de los bloques de código deben ser de dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach. No se debe usar el tabulador; ya que este puede variar según la computadora o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deben estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Ejemplo:

```
for (int i = 0; i < tokens.size(); i++)
{
  if(tokens.get(i).getPalabra().equals(analisis.getLex().get(i).getPalabra()))
  {
    tokens.get(i).setCatGramatical(analisis.getLex().get(i).getCatGramatical());
    tokens.get(i).setLexema(convertirAMayucula(convertirSingular(analisis.getLex().get(i).getLexema())));
  }
}
```

- Para comentar el código se utilizará, los comentarios de implementación, delimitados por /*...*/, y //.

3.2. Pruebas de Software

Las pruebas de software constituyen una fase del proceso de desarrollo de un software centrada en la calidad, fiabilidad y robustez del mismo; dentro del contexto o escenario previsto para ser utilizado. Las mismas permiten detectar la presencia de errores que generen salidas o comportamientos inapropiados durante su ejecución.

Para el desarrollo de las pruebas se tienen en cuenta un conjunto de estrategias a seguir, y de esta forma lograr la calidad requerida y el cumplimiento de los objetivos. Según (Pressman, 2005) *«la estrategia de prueba que elige la mayor parte de los equipos de software se ubica entre estos dos extremos. Toma un enfoque incremental de las pruebas; inicia con la prueba de unidades individuales del programa, pasa a pruebas diseñadas para facilitar la integración de las unidades, y culmina con pruebas que realizan sobre el sistema construido».* El equipo de desarrollo adoptó esta estrategia, pues bajo este enfoque, se realizaron pruebas de unidad al subsistema, las cuales permitieron obtener cada unidad funcional independiente libre de errores.

Pruebas Unitarias

Las pruebas unitarias permiten probar, como su nombre lo indica, cada unidad independiente del software. Su objetivo es el aislamiento de partes del código y la demostración de que estas partes no contienen errores. Los dos grandes grupos de pruebas unitarias existentes son las pruebas de Caja Negra y las pruebas de Caja Blanca.

Pruebas de Caja Blanca

Para la validación de la solución propuesta se realizaron pruebas de Caja Blanca debido a que estas revisan la parte interna del software, específicamente sobre el código fuente. Estas pruebas se basan en

el examen minucioso de los detalles procedimentales, donde se comprueban los caminos lógicos del subsistema, al generar casos de prueba que ejerciten las estructuras condicionales y los bucles.

Según (Pressman, 2005) *«al emplear los métodos de prueba de caja blanca, el ingeniero del software podrá derivar casos de prueba que 1) garanticen que todos las rutas independientes dentro del módulo se han ejercitado por lo menos una vez, 2) ejerciten los lados verdadero y falso de todas las decisiones lógicas, 3) ejecuten todos los bucles en sus límites y dentro de sus límites operacionales, y 4) ejerciten estructuras de datos internos para asegurar su validez. Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad»*.

Dentro de las pruebas de caja blanca, se ejecutaron casos de pruebas tanto de forma manual, como automática, para garantizar la veracidad de los resultados obtenidos.

Para realizar el proceso de prueba de forma manual, se decidió por el equipo de desarrollo utilizar la técnica de camino básico, y de forma automática la herramienta EclEmma, la cual brinda un conjunto de librerías que se integran fácilmente al IDE de desarrollo seleccionado: Eclipse. A continuación se describe en qué consiste la técnica de prueba de caja blanca seleccionada, así como los casos de pruebas aplicados y su comparación con los resultados obtenidos de la herramienta EclEmma.

Técnica de camino básico

La selección de esta técnica está basada en las ventajas que ofrece con respecto a otras técnicas, dado que el número mínimo requerido de pruebas se conoce por adelantado y por tanto el proceso de prueba se puede planear y supervisar en mayor detalle con respecto a las restantes técnicas. Por otra parte, esta técnica asegura que los casos de prueba diseñados permitan que todas las sentencias del programa sean ejecutadas al menos una vez y que las condiciones sean probadas tanto para su valor verdadero como falso.

Los pasos a realizar para aplicar esta técnica son:

- Representar el programa en un grafo de flujo: se utiliza para representar el flujo de control lógico de un programa.
- Calcular la complejidad ciclomática: el valor calculado define el número de rutas independientes en el conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse, lo cual asegura que todas las instrucciones se hayan ejecutado por lo menos una vez.
- Determinar el conjunto básico de rutas independientes: es cualquier ruta del programa que ingresa por lo menos un nuevo conjunto de instrucciones de procesamiento o una nueva condición.
- Derivar los casos de prueba que fuerzan la ejecución de cada camino.

A continuación se muestra el caso de prueba guiado por los pasos anteriores al método **extraerConceptos,** el cual se localiza en la clase controladora **ExtraerConceptos**.

El objetivo de este método es determinar dada una lista de tokens con su CG y RM las frases conceptuales del texto. Se selecciona este método teniendo en cuenta la importancia que representa para el resultado y la implementación del algoritmo.

A continuación, en la figura 20 se muestra el código fuente referente al método descrito anteriormente y en la figura 21 y 22 se aplica la Técnica de Camino Básico al mismo.

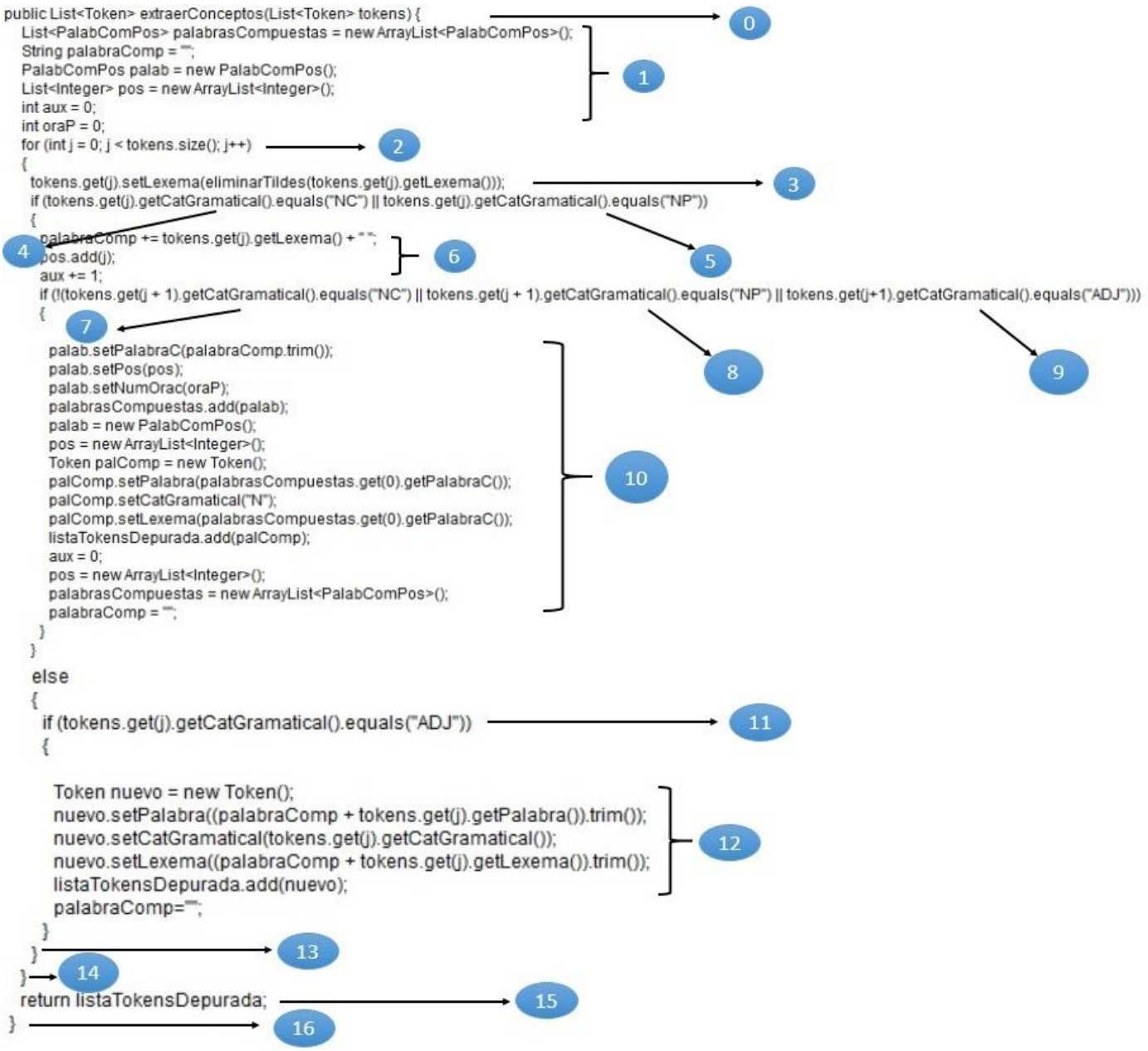
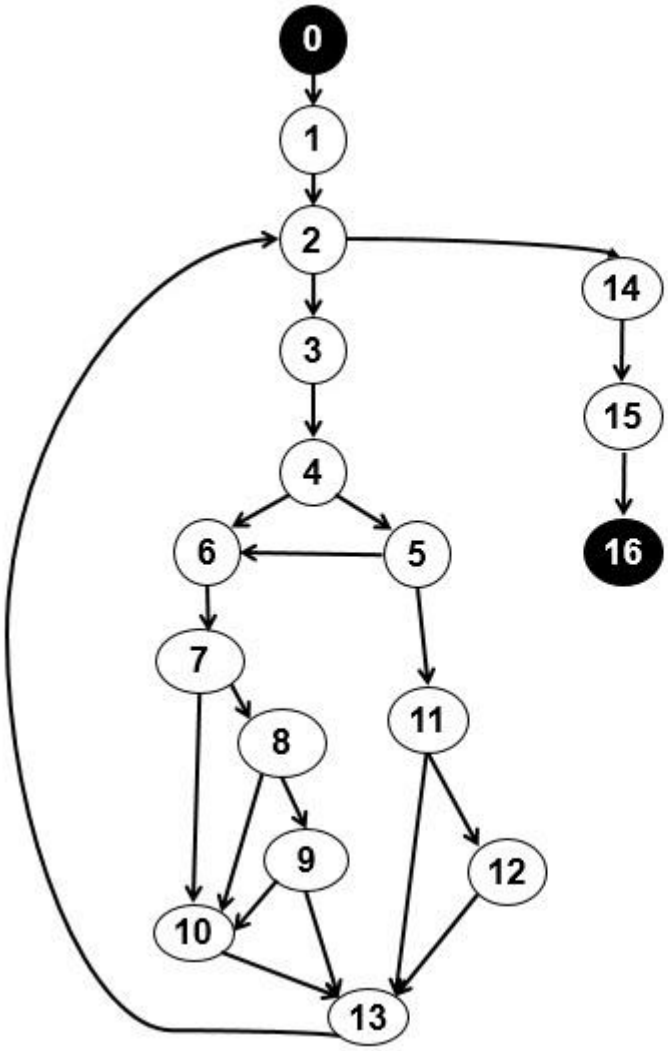


Figura 20: Código fuente correspondiente al método extraerConceptos de la clase ExtraerConceptos



Complejidad Ciclomática

A=26(aristas)

N=17(nodos)

V(G)= A-N+2

V(G)= 23-17+2

V(G)= 8

P=7(nodos predicados)

V(G)= P+1

V(G)=7+1

V(G)=8

R=8(Regiones del grafo)

V(G)=R

V(G)=8

Figura 21: Gráfica de flujo y Complejidad ciclomática correspondiente al método extraerConceptos

Conjunto de Rutas Independientes

Ruta 1: 0-1-2-14-15-16
Ruta 2: 0-1-2-3-4-6-7-10-13-2-14-15-16
Ruta 3: 0-1-2-3-4-5-6-7-10-13-2-14-15-16
Ruta 4: 0-1-2-3-4-6-7-8-10-13-2-14-15-16
Ruta 5: 0-1-2-3-4-6-7-8-9-10-13-2-14-15-16
Ruta 6: 0-1-2-3-4-6-7-8-9-13-2-14-15-16
Ruta 7: 0-1-2-3-4-5-11-13-2-14-15-16
Ruta 8: 0-1-2-3-4-5-11-12-13-2-14-15-16

Figura 22: Rutas linealmente independientes correspondientes al método extraerConceptos

A continuación se diseñan los casos de prueba que cubren los caminos independientes presentados.

Caso de prueba del camino 1:

Entrada: recibe como parámetro una Lista de Tokens vacía

Resultado Esperado: -

Objetivo: se garantiza que se cumpla la condición de que la lista de tokens sea cero.

Caso de prueba del camino 2:

Entrada: recibe como parámetro una Lista de Tokens con su CG y RM, siendo estos Tokens:

educación NC educación

enseña VLfin enseñar

Resultado Esperado: Educación.

Objetivo: se garantiza que se cumpla la condición de que la CG del token actual sea un NC y la del siguiente token distinta de NC.

Caso de prueba del camino 3:

Entrada: recibe como parámetro una Lista de Tokens con su CG y RM, siendo estos Tokens:

UCI NP UCI

enseña VLfin enseñar

Resultado Esperado: UCI.

Objetivo: se garantiza que se cumpla la condición de que la CG del token actual sea un NP y la del siguiente tokens distinta de NC.

Caso de prueba del camino 4:

Entrada: recibe como parámetro una Lista de Tokens con su CG y RM, siendo estos Tokens:

educación NC educación

enseña VLfin enseñar

Resultado Esperado: UCI.

Objetivo: se garantiza que se cumpla la condición de que la CG del token actual sea NC y la del siguiente token sea distinta de NP.

Caso de prueba del camino 5:

Entrada: recibe como parámetro una Lista de Tokens con su CG y RM, siendo estos Tokens:

educación NC educación

enseña VLfin enseñar

Resultado Esperado: educación

Objetivo: se garantiza que se cumpla la condición de que la CG del token actual sea NC y la del siguiente token sea distinta de NP.

Caso de prueba del camino 6:

Entrada: recibe como parámetro una Lista de Tokens con su CG y RM, siendo estos Tokens:

UCI NC UCI

grande ADJ grande

Resultado Esperado: concepto UCI.

Objetivo: se garantiza que se cumpla la condición de que la CG del token actual sea un NC y la del siguiente token sea ADJ.

Caso de prueba del camino 7:

Entrada: recibe como parámetro una Lista de Tokens con su CG y RM, siendo estos Tokens:

superior ADJ superior

Resultado Esperado: concepto grande.

Objetivo: se garantiza que no se cumpla la condición de que la CG del token actual sea un NC o NP y que se cumpla que el token sea un ADJ.

Caso de prueba del camino 8:

Entrada: recibe como parámetro una Lista de Tokens con su CG y RM, siendo estos Tokens:

educar VLfin educar

enseña VLfin enseñar

Resultado Esperado: -

Objetivo: se garantiza que no se cumpla la condición de que la CG del token actual sea un NC, NP o ADJ.

Una vez realizado el procedimiento de forma manual, se procede a utilizar la herramienta EclEmma al método descrito anteriormente para comprobar los resultados. EclEmma es una herramienta que permite realizar pruebas de Cobertura de Sentencias, las cuales describen casos de prueba suficientes para que cada sentencia en el programa se ejecute, al menos, una vez.

En las pruebas automáticas se utilizaron los mismos casos de pruebas que de la forma manual, para comprobar los resultados. En cada caso de la forma automática, quedará marcada la cobertura del código de la siguiente manera:

- En color verde, las líneas de código que han sido ejecutadas.
- En color amarillo, las líneas de código que han sido ejecutadas pero no totalmente.
- En color rojo, las líneas de código que no han sido ejecutadas.

En las figuras 23 y 24 se muestran los resultados de los casos de pruebas 1 y 2 aplicados de forma automática; los restantes casos de pruebas se encuentran en el Anexo 2.

```
@Test
public void extraerConceptos() {
    List<PalabComPos> palabrasCompuestas = new ArrayList<PalabComPos>();
    String palabraComp = "";
    PalabComPos palab = new PalabComPos();
    List<Integer> pos = new ArrayList<Integer>();
    int aux = 0;
    int oraP = 0;
    for (int j = 0; j < tokens.size(); j++) {
        tokens.get(j).setLexema(eliminarTildes(tokens.get(j).getLexema()));
        if (tokens.get(j).getCatGramatical().equals("NC") || tokens.get(j).getCatGramatical().equals("NP")){
            palabraComp += tokens.get(j).getLexema() + " ";
            pos.add(j);
            aux += 1;
            if (!(tokens.get(j + 1).getCatGramatical().equals("NC") || tokens.get(j + 1).getCatGramatical().equals("NP")
                    || tokens.get(j+1).getCatGramatical().equals("ADJ"))){
                palab.setPalabraC(palabraComp.trim());
                palab.setPos(pos);
                palab.setNumOrac(oraP);
                palabrasCompuestas.add(palab);
                palab = new PalabComPos();
                pos = new ArrayList<Integer>();
                Token palComp = new Token();
                palComp.setPalabra(palabrasCompuestas.get(0).getPalabraC());
                palComp.setCatGramatical("N");
                palComp.setLexema(palabrasCompuestas.get(0).getPalabraC());
                listaTokensDepurada.add(palComp);
                aux = 0;
                pos = new ArrayList<Integer>();
                palabrasCompuestas = new ArrayList<PalabComPos>();
                palabraComp = "";
            }
        }
        else{
            if (tokens.get(j).getCatGramatical().equals("ADJ")) {
                Token nuevo = new Token();
                nuevo.setPalabra((palabraComp + tokens.get(j).getPalabra()).trim());
                nuevo.setCatGramatical(tokens.get(j).getCatGramatical());
                nuevo.setLexema((palabraComp + tokens.get(j).getLexema()).trim());
                listaTokensDepurada.add(nuevo);
                palabraComp="";
            }
        }
    }
}
```

Figura 23: Resultado del caso de prueba 1

```
@Test
public void extraerConceptos() {
    Token t = new Token();
    t.setCatGramatical("NC");
    t.setLexema("Educacion");
    t.setPalabra("Educacion");
    tokens.add(t);
    Token t1 = new Token();
    t1.setCatGramatical("VLfin");
    t1.setLexema("enseñar");
    t1.setPalabra("enseña");
    tokens.add(t1);

    List<PalabComPos> palabrasCompuestas = new ArrayList<PalabComPos>();
    String palabraComp = "";
    PalabComPos palab = new PalabComPos();
    List<Integer> pos = new ArrayList<Integer>();
    int aux = 0;
    int oraP = 0;
    for (int j = 0; j < tokens.size(); j++) {
        tokens.get(j).setLexema(eliminarTildes(tokens.get(j).getLexema()));
        if (tokens.get(j).getCatGramatical().equals("NC") || tokens.get(j).getCatGramatical().equals("NP")){
            palabraComp += tokens.get(j).getLexema() + " ";
            pos.add(j);
            aux += 1;
            if (!(tokens.get(j + 1).getCatGramatical().equals("NC") || tokens.get(j + 1).getCatGramatical().equals("NP")
                    || tokens.get(j+1).getCatGramatical().equals("ADJ"))){
                palab.setPalabraC(palabraComp.trim());
                palab.setPos(pos);
                palab.setNumOrac(oraP);
                palabrasCompuestas.add(palab);
                palab = new PalabComPos();
                pos = new ArrayList<Integer>();
                Token palComp = new Token();
                palComp.setPalabra(palabrasCompuestas.get(0).getPalabraC());
                palComp.setCatGramatical("N");
                palComp.setLexema(palabrasCompuestas.get(0).getPalabraC());
                listaTokensDepurada.add(palComp);
                aux = 0;
                pos = new ArrayList<Integer>();
                palabrasCompuestas = new ArrayList<PalabComPos>();
                palabraComp = "";
            }
        }
        else{
            if (tokens.get(j).getCatGramatical().equals("ADJ")) {
                Token nuevo = new Token();
                nuevo.setPalabra((palabraComp + tokens.get(j).getPalabra()).trim());
                nuevo.setCatGramatical(tokens.get(j).getCatGramatical());
                nuevo.setLexema((palabraComp + tokens.get(j).getLexema()).trim());
                listaTokensDepurada.add(nuevo);
                palabraComp="";
            }
        }
    }
}
```

Figura 24: Resultado del caso de prueba 2

La ejecución de los Casos de Prueba de Caja Blanca en el método seleccionado, fue satisfactorio en todos los casos. Se logró cubrir todos los caminos y con ellos todas las aristas del grafo. La complejidad ciclomática es baja lo que permite un mayor entendimiento, facilidad de prueba, modificación y reutilización. Además, se comprobó que los resultados obtenidos con la técnica de camino básico y la herramienta EclEmma fueron los esperados, lo que demuestra calidad y confiabilidad del subsistema.

3.3. Validación de la variable de la investigación

La investigación desarrollada plantea como hipótesis: *la utilización de un subsistema de análisis semántico de texto en el proceso de análisis de datos de la concepción de SASPED, permitirá extraer información en los textos de las respuestas a las preguntas abiertas, para ser utilizadas en la confección de las estrategias individuales de Superación Pedagógica*. A continuación se evalúa la variable **información** con el uso del Subsistema Análisis Semántico de Texto para SASPED, a partir de la operacionalización realizada en el primer epígrafe del segundo capítulo de esta memoria. Para ello se utiliza como método científico el experimento.

Evaluación

En este apartado se presenta la base de conocimiento que se utilizó en la evaluación del subsistema propuesto. Se ofrecen las medidas para conocer el grado de fiabilidad de los resultados. Y se analizan los resultados obtenidos.

Bases de Conocimiento

Para el desarrollo del experimento se utilizó como base de conocimiento una ontología:

Educación: Ontología relacionada con definiciones de la educación y las Tecnologías de la Información y las Comunicaciones (TIC) en idioma español.

Conociendo el dominio de la base de conocimiento, se utilizaron 6 posibles respuestas a preguntas abiertas para realizar el experimento.

Medidas

Para la evaluación de las preguntas se tuvieron en cuenta los siguientes indicadores:

Información Explícita:

- Precisión (P): ***P= CO/CV***
- Cobertura (C): ***C= CV/CE***

Donde **CE** corresponde a la cantidad de conceptos extraídos del texto, **CV** la cantidad de conceptos válidos que se recobraron y **CO** a la cantidad de conceptos asociados a la ontología.

Información Implícita:

Para el cálculo de la precisión y la cobertura de la información implícita, es necesario un experto y es calculada de la siguiente forma:

- Precisión (P): ***P= RP/RV***
- Cobertura (C): ***C= RV/TR***

Donde **RV** corresponde al número de resultados válidos de la respuesta, **RP** a la cantidad de resultados más precisos y **TR** al total de resultados obtenidos.

Resultados

A partir de la muestra seleccionada con respecto a las respuestas de la pregunta abierta realizada en el diagnóstico de la ESPC-UCI (Ver Anexo 3), se muestran en la tabla 12 los resultados de estas respuestas al ser procesadas por el subsistema de análisis semántico de texto, además del cálculo de la precisión y la cobertura de la información extraída.

La precisión y la cobertura significan el nivel de exactitud en la respuesta. Las respuestas más exactas son las que su precisión y cobertura está más cercana a uno.

A continuación se muestra en la figura 25 el cálculo de la precisión y cobertura de la información explícita del texto procesado que se muestra en la figura 18 del acápite de implementación.

Precisión (P): P= CO/CV
Cobertura (C): C= CV/CE

CO=8	P=8/9	C=9/12
CV=9	P=0.88	C=0.75
CE=12	**P=88%**	**C=75%**

Figura 25: Cálculo de la precisión y cobertura de la información explícita

Del total de 6 respuestas tomadas como muestra, se obtuvo un promedio de precisión y cobertura de la información explícita del 84.5% y 81.8% respectivamente; y de la información implícita un 83% de precisión y un 82.7% de cobertura, lo que demuestra que:

- Se obtiene información explícita e implícita con un nivel de precisión y cobertura alto, lo que refleja exactitud en las respuestas.

Los resultados que se muestran a continuación en la tabla 11 pertenecen a la comparación de los niveles de extracción de información inicial y actual del proceso de análisis de datos de la concepción de SASPED en lo referente a las preguntas abiertas.

Tabla 11: Comparación de los niveles de extracción de información

Estado	Precisión	Cobertura	Precisión	Cobertura
	Información Explícita		Información Implícita	
Inicial	Nulo	Nulo	Nulo	Nulo
Actual	Alta	Alta	Alta	Alta

Una vez analizados los resultados y teniendo en cuenta lo anterior, se puede destacar que se obtiene la información explícita e implícita de los textos en las respuestas a las preguntas abiertas con exactitud. La transformación ocurrida en el proceso de análisis de datos de la concepción de SASPED, permite constatar la validez de la hipótesis planteada y responder al problema científico identificado.

Tabla 12: Resultados generados por el subsistema de análisis semántico de texto

No.	Respuestas del subsistema de análisis semántico de texto	Precisión	Cobertura	Precisión	Cobertura
		Información Explícita		Información Implícita	
1	Entorno Virtual Enseñanza Aprendizaje, determinado por principios didácticos, facilitan el trabajo colectivo, el intercambio socio cultural, el autoaprendizaje, ofrecen servicio tecnológico, es una forma de tecnología educativa, propicia la comunicación, constituye un contexto educativo y es un espacio virtual. TIC, permiten tratamiento de información, registro de información, producción de información, presentación de información, almacenamiento de información, adquisición de información. Autoaprendizaje es un aprendizaje. Educación Superior, usa las TIC.	88%	75%	85%	80%
2	TIC, permiten tratamiento de información, registro de información, producción de información, presentación de información, almacenamiento de información, adquisición de información. Hipermedia, contiene video, texto, sonido, imagen, animación y es una TIC. Multimedia, integra hipertexto, hipermedia y es una TIC. Servicio Tecnológico, son para la comunicación, sustentado por las TIC.	60%	80%	86%	88%
3	Didáctica, es una ciencia pedagógica. Contexto Educativo, es un contexto. Profesor, recibe una formación básica, es una persona. Autoaprendizaje, es un aprendizaje. Cultura Aprendizaje, es una capacidad.	71%	87%	75%	67%
4	Medio Enseñanza Aprendizaje, es un mediador de la enseñanza aprendizaje, permite desarrollo de habilidad, es un recurso tecnológico. Actividad Educativa, es una actividad formativa. Aprendizaje, tiene un carácter participativo, interactivo y colaborativo. Proceso Docente Educativo, es una actividad formativa.	100%	83%	87%	100%

No.	Respuestas del subsistema de análisis semántico de texto	Precisión	Cobertura	Precisión	Cobertura
		Información Explícita		Información Implícita	
5	TIC, permiten tratamiento de información, registro de información, producción de información, presentación de información, almacenamiento de información, adquisición de información. Habilidad, es una capacidad. Habito, es una práctica adquirida. Profesor, recibe formación básica y es una persona. Asimilación, es un proceso del pensamiento. Recursos Tecnológicos, posibilitan enseñanza aprendizaje.	88%	80%	80%	83%
6	Aprendizaje, tiene un carácter participativo, interactivo y colaborativo. Profesor, recibe formación básica y es una persona. Información, es brindada por servicio tecnológico. Estudiante, es una persona. Estrategia Enseñanza, hace efectivo el proceso enseñanza aprendizaje y son técnicas. Autoaprendizaje, es un aprendizaje.	100%	86%	85%	78%

Conclusiones del Capítulo

- La implementación del algoritmo propuesto, integra en seis operaciones secuenciales el procesamiento de análisis semántico a un texto en lenguaje natural, lo que permite transitar por las etapas de pre-procesamiento, descubrimiento y obtención de los resultados con un mayor procesamiento de la información contenida en el texto inicial.
- Existen dos grandes grupos para las pruebas de software, siendo las pruebas de Caja Blanca las utilizadas en el subsistema propuesto, por permitir: reducir el porciento de representación de errores existentes, el examen minucioso de los detalles procedimentales y comprobar los caminos lógicos del algoritmo.
- La aplicación de la **técnica camino básico** y el uso de la herramienta **EclEmma**, permitió corroborar los resultados respecto a los casos de pruebas que por separado se obtuvieron en cada una de ellas.
- La evaluación de la variable considerada en la investigación utilizando la técnica de experimento, demostró que la solución desarrollada transforma el proceso de análisis de datos de la concepción de SASPED, en lo referente a la extracción de información en los textos de las respuestas a las preguntas abiertas.

CONCLUSIONES FINALES

La investigación desarrollada y los resultados obtenidos permiten a los autores plantear las siguientes conclusiones finales:

- El estudio de los sistemas existentes para realizar el análisis semántico a un texto, demuestra que los mismos no brindan solución al problema de la investigación, por lo que fue necesario una propuesta de herramienta de análisis semántico de texto, desarrollada sobre el área de conocimiento Minería de Texto y que utiliza una ontología como base de conocimiento.
- En el proceso de análisis de datos de la concepción de SASPED es necesario el análisis de las respuestas a las preguntas abiertas para obtener información explícita e implícita, y ser utilizada en la confección de las estrategias individuales de Superación Pedagógica.
- Se desarrolló la investigación con la metodología **OpenUP** y la arquitectura **Flujo de Datos** porque responden a necesidades y condiciones objetivas (tiempo limitado, recursos disponibles, equipo pequeño) y personológicas (amplias capacidades y habilidades, organización propia) del equipo de desarrollo. Además, la arquitectura es la definida porque provee una estructura para sistemas que procesan flujo de datos, respondiendo esto al diseño de la solución propuesta.
- El experimento para la validación de la variable permitió valorar la transformación del proceso de análisis de datos de la concepción de SASPED y arribar a nuevos conocimientos, al modificarse la extracción de información en los textos de las respuestas a las preguntas abiertas a partir del subsistema de análisis semántico de texto.

RECOMENDACIONES

A partir del estudio realizado en la presente investigación y teniendo en cuenta las experiencias adquiridas a lo largo de su desarrollo, los autores realizan las siguientes recomendaciones:

- Desambiguar el sentido de las palabras identificadas en el texto, para asignar el significado correcto a la palabra y así utilizarla en el contexto particular del texto, logrando aumentar el nivel de precisión y cobertura de la información extraída.
- Permitir el aprendizaje automático de la ontología para enriquecer su conocimiento y así elevar el nivel de extracción de información del texto.
- Valorar la implementación de un análisis de sentimientos para determinar la polaridad de las respuestas y así analizar la tendencia de las opiniones y sentimientos de las personas encuestadas, para elevar el nivel de precisión y cobertura de la información extraída.

REFERENCIAS BIBLIOGRÁFICAS

- **Aguilar, O. (2009).** Cruzadas audiovisuales: metodología heurística para un análisis semántico–cognitivo del spot electoral. *Comunicación y sociedad*, 63-100. Recuperado el 10 de 11 de 2013, de http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-252X2009000200004
- **Alcocer, D. E., & Ortiz, F. R. (2013).** Diseño e Implementación del Prototipo de un Sistema Computarizado Distribuido Orientado a la Utilización de los Servicios en un Campus Universitario a nivel local con capacidad para 2000 a 5000 estudiantes. . Quito, Ecuador.
- **Alizegui, A. (2013).** *Proyecto de Ley, la Legislatura de la Provincia de entre Ríos Sanciona con Fuerza de Ley. "Prohibición de retención de Documentación oficial del Alumno".* México.
- **Almeira, A., & Cavenago, V. (2007).** *Arquitectura de Software: Estilos y Patrones.* Argentina.
- **Avison, D., & Fitzgerald, G. (1995).** *Information Systems Development: Methodologies, Techniques and Tools.* Paul & Company Publishers Consortium.
- **Banerjee, S. (2002).** *Adapting the lesk algorithm for word sense disambiguation using wordnet.* Minnesota: Department of Computing Science.
- **Berners-Lee, T., Hendler, J., & Lassila, O. (2001).** *The Semantic Web. Scientific American.*
- **Borst, P. (1997).** *Construction of Engineering Ontologies for Knowledge Sharing and Reuse.* Tweente University.
- **Botta, E., & Cabrera, J. (2007).** Minería de textos: una herramienta útil para mejorar la gestión del bibliotecario en el entorno digital. *SciELO*.
- **Chiavenato, I. (2006).** *Introducción a la Teoría General de la Administración* (Séptima ed.). McGraw-Hill Interamericana.
- **Ciudad, F. A., Díaz, T., Blanco, S. M., Puentes, Ú., López, J. F., & Martínez, O. L. (2013).** Propuesta de Estrategia de Superación Pedagógica del Claustro. La Habana, Cuba.
- **Codina,, L., & Pedraza, R. (2011).** Tesauros y ontologías en sistemas de información documental. *El profesional de la información*, 555-563.
- **Dávila, H., Fernández, A., Gutiérrez, Y., Muñoz, R., Montoyo, A., & Vázquez, S. (2012).** Método de Extracción de Información Semántica en ontologías. Matanza, Cuba.
- **DRAE. (2014).** *Diccionario de la lengua española*. Obtenido de Diccionario de la lengua española: http://www.rae.es/recursos/diccionarios/drae

- **Feldman, R., Fresko, M., Kinar, Y., Lindell, Y., Liphstat, O., Rajman, M., . . . Zamir, O. (1998).** Text Mining at the Term Level. *2nd European Symposium Principles of Data Mining and Knowledge Discovery.*
- **Fernández, F. (2011).** Integración de métodos para la desambiguación del sentido de las palabras en el contexto del procesamiento del lenguaje natural. La Habana, Cuba.
- **Flores, F. (2011).** *Aplicación de METHONTOLOGY para la Construcción de una Ontología en el Domino de la Microbiología. Caso de Estudio: Identificación de Bacilos Gram Negativos no Fermentadores de la Glucosa (BGNNF).* Caracas, Venezuela.
- **Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2003).** *Patrones de diseño. Elementos de software orientado a objetos reutilizable.* Madrid: Pearson educación. S.A.
- **García, G., & Medina, J. (2010).** *Procesamineto de expresiones valorativas para la sumarización de opiniones.* La Habana: Reporte Técnico.
- **Gruber, T. (1993).** *A Translation Approach to Portable Ontology Specifications.* Stanford: Stanford University.
- **Hearst, M. (1999).** Untangling text data mining. *Proceedings of ACL'99: the 37th Annual Meeting of the Association For Computational Linguistic ACL.*, (págs. 3-10). Maryland.
- **Hennig, L., Umbrath, W., & Wetzker, R. (2008).** An Ontology-based Approach to Text Summarization. *EEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.* Princenton: IEEE.
- **Hernández, F., García-Sanz, M., & Maquilón, J. (2009).** Análisis de los datos cualitativos.
- **Hernández, R. A., & Coello, S. (2011).** *El proceso de investigación científica.* La Habana: Editorial Universitaria del Ministerio de Educación Superior.
- **Hernández, R., Fernández-Collado, C., & Baptista, P. (2006).** *Metodología de la Investigación* (Cuarta ed.). México: McGraw-Hill.
- **Hu, P., He, T., Ji, D., & Wang, M. (2004).** A Study of Chinese Text Summarization Using Adaptive Clustering of Paragraphs. *Proceedings of the Fourth International Conference on Computer and Information Technology.*
- **IFAI-UCI-MES. (2010).** *Informe Final de Evaluación Institucional del MES a la UCI[Documento interno UCI].* La Haban: Universidad de las Ciencias Informáticas, Cuba.
- **Jacobson, I., Booch, G., & Rumbaugh, J. (2000).** *El Proceso Unificado de Desarrollo de software. La guía completa del Proceso Unificado escrita por sus creadores.* Madrid: Pearson Educación. S.A.

- **Kodratoff, Y. (1999).** Knowledge Discovery in Texts: A Definition, and Applications. *Proceedings of the 11th International Symposium on Foundations of Intelligent Systems* (págs. 16-29). Springer-Verlag.
- **Kowata, J., Cury, D., & Beores, M. (2010).** Concept Maps Core Elements Candidates Recognition From Texts. *Proc. of Fourth International Conference on Concept Mapping.* Viña del Mar, Chile.
- **Kutschera, F. (1979).** Semántica en DF 11, Filosofía del lenguaje Gredos. Obtenido de http://mercaba.org/VocTEO/S/semantica.htm
- **Larman, C. (1999).** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México: Prentice Hall.
- **Leiva-Mederos, A. A. (2012).** Texminer: Un Modelo para el Resumen Automático y la Desambiguación de Textos Científicos en el Dominio de Ingeniería de Puertos y Costas. La Habana, Cuba.
- **Leiva-Mederos, A., Domínguez-Fernández, S., & Senso, J. A. (2012).** PuertoTex: un software de minería textual para la creación de resúmenes automáticos en el dominio de ingeniería de puertos y costas basado en ontologías. *TransInformação, 24*(2), 103-115. Obtenido de http://eprints.rclis.org/18519/
- **Lesk, M. (1986).** Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. *5th annual international conference on Systems documentation.* Ontario.
- **Maduro, R., & Rodríguez, J. (2008).** Degustando el sabor de los Datos Cualitativos. *Actualidades Investigativas en Educación*, 24.
- **Mamani, N. (2010).** *Web Semántica: Ontologías, Agentes de Software y Servicios Web Semánticos.*
- **Marina, J. L. (2008).** FastUMLS: Extracción de conceptos en textos biomédicos. *Proyecto de Fin de Máster en Ingeniería de Computadores*. Madrid, España: Universidad Complutense de Madrid.
- **Mayz, C. (2009).** ¿Cómo desarrollar, de manera comprensiva, el análisis cualitativo de los datos? *EDUCERE • Artículos arbitrarios*, 12.
- **Mejía, J. (2011).** Problemas centrales del análisis de datos cualitativos. *Revista Latinoamericana de Metodología de la Investigación Social.*, 14.
- **Mira, J. J., Gómez, J., Blaya, I., & García, A. (2006).** La Gestión por Procesos. Alicante, España.

- **Montes y Gómez, M. (2001).** *Minería de texto: Un nuevo reto computacional.* México: Centro de Investigación en Computación, Instituto Politécnico Nacional. Obtenido de http://ccc.inaoep.mx/~mmontesg/publicaciones/2001/MineriaTexto-md01.pdf
- **Ning, H., & Shihan, D. (2006).** Structure-Based Ontology Evaluation. *International Conference on e-business on Engineering.* Computer Society, IEEE.
- **Noy, N., & McGuinness, D. (2005).** Desarrollo de Ontologías-101: Guía para crear tu primera ontología.
- **Onieva, J. L. (1990).** *Curso de comunicación activa.* Plaza Mayor.
- **Piñero, A. (2010).** *El análisis semántico: métodos literarios aplicados al estudio del Nuevo Testamento (III).* Obtenido de Cristianismo e Historia: http://www.tendencias21.net/crist/El-analisis-semantico-metodos-literarios-aplicados-al-estudio-del-Nuevo-Testamento-III-200-48_a689.html
- **Pressman, R. (2005).** *Ingeniería de Software. Un enfoque práctico. Sexta Edición.* Mc Graw Hill.
- **Proenza, Y., & Pérez, A. (2012).** OntoCatMedia: ontología para la búsqueda y clasificación automática de medias audiovisuales. *Ciencias de la Información*, 49 - 54.
- **Ramos, E., & Nuñez, H. (2007).** *ONTOLOGÍAS: componentes, metodologías, lenguajes, herramientas y aplicaciones.* Caracas: Lecturas en Ciencias de la Computación.
- **Reyes, F. (2012).** Análisis de datos cualitativos en los trabajos de investigación. Recuperado el 15 de 10 de 2013, de http://periplosenred.blogspot.com/2012/03/analisis-de-datos-cualitativos-en-los.html
- **Rodríguez, A., & Simón, A. (2013).** Método para la extracción de información estructurada desde textos. *Revista Cubana de Ciencias Informáticas*, 55-67.
- **Rodríguez, C., Lorenzo, O., & Herrera, L. (2005).** Teoría y práctica del análisis de datos cualitativos. Proceso general y criterios de calidad. *Revista Internacional de Ciencias Sociales y Humanidades, XV*(2).
- **Rodríguez, G., Gil, J., & García, E. (1996).** *Metodología de la Investigación Cualitativa.*
- **Samper, J. (2005).** *Ontologías para Servicios Web Semánticos de Información de Tráfico: Descripción y Herramientas de explotación.* Valencia: Universitat de València.
- **Sánchez, B., & Jaimes, R. (1985).** *Entropía curricular. Reto para la educación del siglo XXI.* Maracay-Venezuela: Editorial Universitaria.
- **Simón, A., Rosete, A., & Ceccaroni, L. (2012).** Método para la generación de ontologías a partir de mapas conceptuales en dominios poco profundos. *II Jornadas sobre Ontologías Web y Semántica*, (págs. 39-46). La Habana.

- **Sommerville, I. (2005).** *Ingeniería de Software* (Séptima ed.). Madrid: Perason Educación, S.A.
- **Studer, R., Benjamins, R., & Fensel, D. (1998).** *Knowledge Engineering: Principles and Methods.*
- ***UNESCO*. (1999).** Recuperado el 2014, de http://www.unesco.org/new/es/unesco/
- **Vallez, M. (2009).** La web semántica y el procesamiento del lenguaje natural. Trea, Gijón.
- **Zhang, X., Cheng, G., & Qu, Y. (2007).** *Ontology Summarization Based on RDF Sentence Graph.* Canadá.

GLOSARIO DE TÉRMINOS

Subsistema: Un subsistema es un sistema que es parte de otro sistema mayor. En otras palabras, un subsistema es un conjunto de elemento interrelacionados que, en sí mismo, es un sistema, pero a la vez es parte de un sistema superior.

Algoritmo: Palabra que viene del nombre del matemático árabe Al-Khwarizmi (780 - 850 aprox.). Es una serie ordenada de instrucciones, pasos o procesos que llevan a la solución de un determinado problema.

Minería de Texto: La extracción no trivial de información implícita, previamente desconocida y potencialmente útil de grandes cantidades de datos texto.

Ontología: Representación formal de un conjunto de conceptos dentro de un dominio específico y de las relaciones entre dichos conceptos.

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

Base de conocimiento: El objetivo de una base de conocimientos es el de modelar y almacenar bajo forma digital un conjunto de conocimiento, ideas, conceptos o datos que permitan ser consultados o utilizados.

Desambiguación del sentido de las palabras (WSD por sus siglas en inglés): Es la habilidad computacional de determinar cuál significado de una palabra debe ser activado debido a su utilización en un contexto particular. Se hace necesario porque una palabra puede ser interpretada de diferentes formas, es decir, posee más de un significado o sentido (fenómeno lingüístico conocido como polisemia). Lo que persigue la WSD es la asignación automática de sentidos a las palabras de un texto.

ANEXOS

Anexo 1: Modelo de dominio de la concepción de SASPED

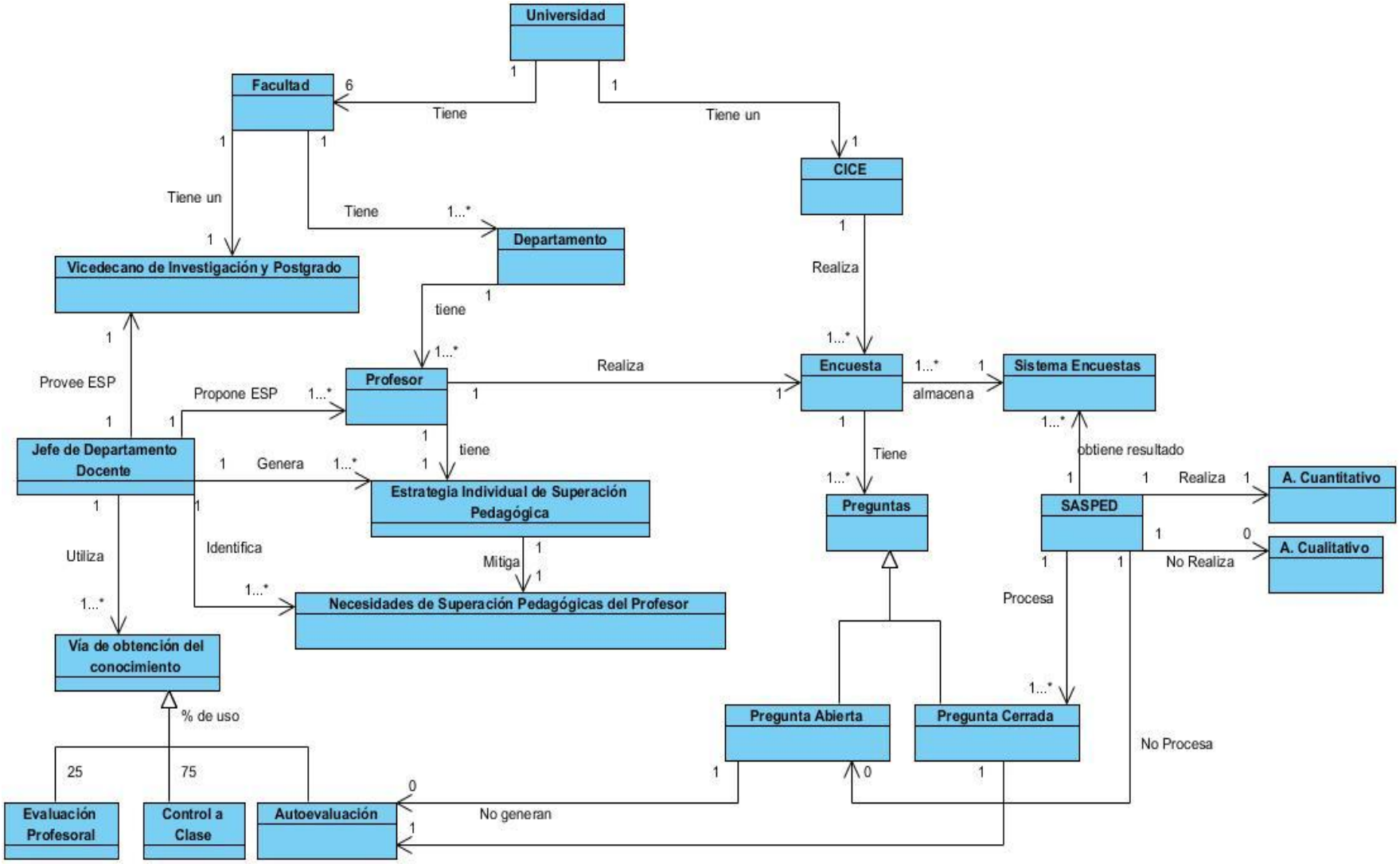


Figura 20. Modelo de dominio de SASPED.

Anexo 2: Casos de pruebas automáticos

```
    @Test
    public void extraerConceptos() {
        Token t = new Token();
        t.setCatGramatical("NP");
        t.setLexema("UCI");
        t.setPalabra("UCI");
        tokens.add(t);
        Token t1 = new Token();
        t1.setCatGramatical("VLfin");
        t1.setLexema("enseñar");
        t1.setPalabra("enseña");
        tokens.add(t1);

        List<PalabComPos> palabrasCompuestas = new ArrayList<PalabComPos>();
        String palabraComp = "";
        PalabComPos palab = new PalabComPos();
        List<Integer> pos = new ArrayList<Integer>();
        int aux = 0;
        int oraP = 0;
        for (int j = 0; j < tokens.size(); j++) {
            tokens.get(j).setLexema(eliminarTildes(tokens.get(j).getLexema()));
            if (tokens.get(j).getCatGramatical().equals("NC") || tokens.get(j).getCatGramatical().equals("NP")){
                palabraComp += tokens.get(j).getLexema() + " ";
                pos.add(j);
                aux += 1;
                if (!(tokens.get(j + 1).getCatGramatical().equals("NC") || tokens.get(j + 1).getCatGramatical().equals("NP")
                        || tokens.get(j+1).getCatGramatical().equals("ADJ"))){
                    palab.setPalabraC(palabraComp.trim());
                    palab.setPos(pos);
                    palab.setNumOrac(oraP);
                    palabrasCompuestas.add(palab);
                    palab = new PalabComPos();
                    pos = new ArrayList<Integer>();
                    Token palComp = new Token();
                    palComp.setPalabra(palabrasCompuestas.get(0).getPalabraC());
                    palComp.setCatGramatical("N");
                    palComp.setLexema(palabrasCompuestas.get(0).getPalabraC());
                    listaTokensDepurada.add(palComp);
                    aux = 0;
                    pos = new ArrayList<Integer>();
                    palabrasCompuestas = new ArrayList<PalabComPos>();
                    palabraComp = "";
                }
            }
            else{
                if (tokens.get(j).getCatGramatical().equals("ADJ")) {
                    Token nuevo = new Token();
                    nuevo.setPalabra((palabraComp + tokens.get(j).getPalabra()).trim());
                    nuevo.setCatGramatical(tokens.get(j).getCatGramatical());
                    nuevo.setLexema((palabraComp + tokens.get(j).getLexema()).trim());
                    listaTokensDepurada.add(nuevo);
                    palabraComp="";
                }
            }
        }
    }
```

Figura 21. Caso de prueba 3.

```
@Test
public void extraerConceptos() {
    Token t = new Token();
    t.setCatGramatical("NP");
    t.setLexema("UCI");
    t.setPalabra("UCI");
    tokens.add(t);
    Token t1 = new Token();
    t1.setCatGramatical("ADJ");
    t1.setLexema("grande");
    t1.setPalabra("grande");
    tokens.add(t1);

    List<PalabComPos> palabrasCompuestas = new ArrayList<PalabComPos>();
    String palabraComp = "";
    PalabComPos palab = new PalabComPos();
    List<Integer> pos = new ArrayList<Integer>();
    int aux = 0;
    int oraP = 0;
    for (int j = 0; j < tokens.size(); j++) {
        tokens.get(j).setLexema(eliminarTildes(tokens.get(j).getLexema()));
        if (tokens.get(j).getCatGramatical().equals("NC") || tokens.get(j).getCatGramatical().equals("NP")){
            palabraComp += tokens.get(j).getLexema() + " ";
            pos.add(j);
            aux += 1;
            if (!(tokens.get(j + 1).getCatGramatical().equals("NC") || tokens.get(j + 1).getCatGramatical().equals("NP")
                    || tokens.get(j+1).getCatGramatical().equals("ADJ"))){
                palab.setPalabraC(palabraComp.trim());
                palab.setPos(pos);
                palab.setNumOrac(oraP);
                palabrasCompuestas.add(palab);
                palab = new PalabComPos();
                pos = new ArrayList<Integer>();
                Token palComp = new Token();
                palComp.setPalabra(palabrasCompuestas.get(0).getPalabraC());
                palComp.setCatGramatical("N");
                palComp.setLexema(palabrasCompuestas.get(0).getPalabraC());
                listaTokensDepurada.add(palComp);
                aux = 0;
                pos = new ArrayList<Integer>();
                palabrasCompuestas = new ArrayList<PalabComPos>();
                palabraComp = "";
            }
        }
        else{
            if (tokens.get(j).getCatGramatical().equals("ADJ")) {
                Token nuevo = new Token();
                nuevo.setPalabra((palabraComp + tokens.get(j).getPalabra()).trim());
                nuevo.setCatGramatical(tokens.get(j).getCatGramatical());
                nuevo.setLexema((palabraComp + tokens.get(j).getLexema()).trim());
                listaTokensDepurada.add(nuevo);
                palabraComp="";
            }
        }
    }
}
```

Figura 22. Caso de prueba 6.

```
@Test
public void extraerConceptos() {
    Token t1 = new Token();
    t1.setCatGramatical("ADJ");
    t1.setLexema("superior");
    t1.setPalabra("superior");
    tokens.add(t1);

    List<PalabComPos> palabrasCompuestas = new ArrayList<PalabComPos>();
    String palabraComp = "";
    PalabComPos palab = new PalabComPos();
    List<Integer> pos = new ArrayList<Integer>();
    int aux = 0;
    int oraP = 0;
    for (int j = 0; j < tokens.size(); j++) {
        tokens.get(j).setLexema(eliminarTildes(tokens.get(j).getLexema()));
        if (tokens.get(j).getCatGramatical().equals("NC") || tokens.get(j).getCatGramatical().equals("NP")){
            palabraComp += tokens.get(j).getLexema() + " ";
            pos.add(j);
            aux += 1;
            if (!(tokens.get(j + 1).getCatGramatical().equals("NC") || tokens.get(j + 1).getCatGramatical().equals("NP")
                    || tokens.get(j+1).getCatGramatical().equals("ADJ"))){
                palab.setPalabraC(palabraComp.trim());
                palab.setPos(pos);
                palab.setNumOrac(oraP);
                palabrasCompuestas.add(palab);
                palab = new PalabComPos();
                pos = new ArrayList<Integer>();
                Token palComp = new Token();
                palComp.setPalabra(palabrasCompuestas.get(0).getPalabraC());
                palComp.setCatGramatical("N");
                palComp.setLexema(palabrasCompuestas.get(0).getPalabraC());
                listaTokensDepurada.add(palComp);
                aux = 0;
                pos = new ArrayList<Integer>();
                palabrasCompuestas = new ArrayList<PalabComPos>();
                palabraComp = "";
            }
        }
        else{
            if (tokens.get(j).getCatGramatical().equals("ADJ")) {
                Token nuevo = new Token();
                nuevo.setPalabra((palabraComp + tokens.get(j).getPalabra()).trim());
                nuevo.setCatGramatical(tokens.get(j).getCatGramatical());
                nuevo.setLexema((palabraComp + tokens.get(j).getLexema()).trim());
                listaTokensDepurada.add(nuevo);
                palabraComp="";
            }
        }
    }
}
```

Figura 23. Caso de prueba 7.

```
@Test
public void extraerConceptos() {
    Token t = new Token();
    t.setCatGramatical("VLfin");
    t.setLexema("educar");
    t.setPalabra("educar");
    tokens.add(t);
    Token t1 = new Token();
    t1.setCatGramatical("VLfin");
    t1.setLexema("enseñar");
    t1.setPalabra("enseña");
    tokens.add(t1);

    List<PalabComPos> palabrasCompuestas = new ArrayList<PalabComPos>();
    String palabraComp = "";
    PalabComPos palab = new PalabComPos();
    List<Integer> pos = new ArrayList<Integer>();
    int aux = 0;
    int oraP = 0;
    for (int j = 0; j < tokens.size(); j++) {
        tokens.get(j).setLexema(eliminarTildes(tokens.get(j).getLexema()));
        if (tokens.get(j).getCatGramatical().equals("NC") || tokens.get(j).getCatGramatical().equals("NP")){
            palabraComp += tokens.get(j).getLexema() + " ";
            pos.add(j);
            aux += 1;
            if (!(tokens.get(j + 1).getCatGramatical().equals("NC") || tokens.get(j + 1).getCatGramatical().equals("NP")
                    || tokens.get(j+1).getCatGramatical().equals("ADJ"))){
                palab.setPalabraC(palabraComp.trim());
                palab.setPos(pos);
                palab.setNumOrac(oraP);
                palabrasCompuestas.add(palab);
                palab = new PalabComPos();
                pos = new ArrayList<Integer>();
                Token palComp = new Token();
                palComp.setPalabra(palabrasCompuestas.get(0).getPalabraC());
                palComp.setCatGramatical("N");
                palComp.setLexema(palabrasCompuestas.get(0).getPalabraC());
                listaTokensDepurada.add(palComp);
                aux = 0;
                pos = new ArrayList<Integer>();
                palabrasCompuestas = new ArrayList<PalabComPos>();
                palabraComp = "";
            }
        }
        else{
            if (tokens.get(j).getCatGramatical().equals("ADJ")) {
                Token nuevo = new Token();
                nuevo.setPalabra((palabraComp + tokens.get(j).getPalabra()).trim());
                nuevo.setCatGramatical(tokens.get(j).getCatGramatical());
                nuevo.setLexema((palabraComp + tokens.get(j).getLexema()).trim());
                listaTokensDepurada.add(nuevo);
                palabraComp="";
            }
        }
    }
}
```

Figura 24. Caso de prueba 8.

Anexo 3: Fragmentos de respuestas de profesores a la pregunta ¿Cuánto considera usted que ha influido las TIC en la educación actual? realizada en el diagnóstico de la ESPC-UCI.

1. Los Entornos Virtuales de Enseñanza Aprendizaje han sido posibles gracias a las TIC. Estos facilitan el autoaprendizaje, el intercambio socio cultural, el trabajo colectivo y la comunicación entre las personas que lo utilizan. Su uso en la educación superior es imprescindible hoy día.
2. El desarrollo de las TIC ha permitido el avance de tecnologías como la hipermedia y la multimedia, diversificando las posibilidades de representar la enseñanza y la información. A su vez, han permitido servicios tecnológicos que han transformado la comunicación a nivel global y la educación a todos los niveles.
3. La didáctica fundamenta la creación de un contexto educativo donde se pone en práctica las leyes y teorías de la pedagogía y demás ciencias de la educación en relación con ella. Además cuando el profesor utiliza las tecnologías educativas desarrolla el autoaprendizaje y aumenta su cultura de aprendizaje del estudiante.
4. Los medios de enseñanza aprendizaje se utilizan en las actividades educativas para enriquecer el aprendizaje. Con estos medios se pueden aplicar los métodos de enseñanza aprendizaje con mayores resultados en el proceso docente educativo.
5. El avance de las TIC ha permitido desarrollar habilidades y hábitos en el trabajo de los profesores. Esto es dado por su asimilación y práctica adquirida con los recursos tecnológicos en su desarrollo profesional.
6. Para el desarrollo del aprendizaje el profesor tiene que buscar diferentes formas de ofrecer la información a sus estudiantes. Por ello recurre al uso de estrategias de enseñanzas y a incentivar el autoaprendizaje. En la actualidad lo hace apoyándose en las TIC y otros recursos tecnológicos disponibles.