
Universidad Nacional de Rosario

Tesina de Licenciatura en Ciencias de la Computación

# Catalogación de Recursos Educativos utilizando Vocabulario Controlado a partir de Ontologías

Autor: Fabricio Mahon

Directora: Cristina Bender Co Directora: Claudia Deco

---

Abril 2016

# Abstract

En este trabajo se presenta un sistema catalogador de documentos mediante el uso de vocabulario controlado representado mediante ontologías. Este sistema puede asistir en la carga de recursos educativos en un repositorio, brindando sus palabras claves. En esta primera propuesta, el dominio de interés es el de las Ciencias de la Computación y se diseñó una ontología y se generó un prototipo que abarca tres áreas de dicho dominio utilizando la metodología Methontology. Se presenta la arquitectura del catalogador y se implementa un prototipo en lenguaje Java, utilizando un extractor de palabras claves del documento, asignación de pesos a los términos en la ontologia a partir de un corpus y una función peso. Y se realiza la catalogación del documento a partir de un vector obtenido del mismo documento. Con respecto a la experimentación y los resutados se utiliza stemming para mejorar precisión de cada catalogación y se experimenta con dos funciones peso. Se obtienen los resultados deseados en la implementación de la propuesta.

# Índice general

# Capítulo 1

# Introducción

El objetivo de este trabajo es diseñar un catalogador automático de recursos educativos mediante la asignación de palabras claves utilizando un vocabulario controlado. La motivación es ayudar a la carga de recursos en un repositorio y mejorar la recuperación de los mismos frente a una búsqueda. En [30] se indaga respecto de las formas de indización utilizadas en los repositorios digitales de acceso abierto en Argentina. Como uno de los resultados de este análisis el 100 % indicó que utiliza vocabulario controlado para la indización pero que ésta es realizada manualmente y ninguno utiliza un software que permita la asignación automática de términos pertenecientes a un vocabulario controlado, de esto se desprende la relevancia del desarrollo aquí planteado. Sin embargo, en estos últimos años se vienen desarrollando sistemas catalogadores automáticos de documentos o sitios web en idioma Inglés. Cada uno presenta características particulares. [2] utilizan ontologías como vocabulario fijo para indexar documentos con grandes cantidades de datos. [14] propone un modelo para la catalogación semi-automática de un sitio web a partir de la creación de un conjunto de metadatos sobre los contenidos de un sitio. [29] presentan una aplicación para clasificar automáticamente textos utilizando una ontología de dominio, basada en el esquema de clasificación Dewey utilizada en bibliotecología. El sistema de clasificación Dewey consiste en un conjunto de clasificación jerárquico de diez categorías y subniveles de las mismas, donde cada nivel se representa con un número compuesto por parte entera y también puede tener una parte decimal. Algunas de las categorías son, 'Filosofía y disciplinas relacionadas' (100), 'Filosofía moderna occidental' (190) 'Arte' (700), 'Artes plásticas; escultura' (730).

En este trabajo se propone una arquitectura de sistema catalogador de documentos utilizando lenguaje controlado por una ontología de dominio. Se implementa el prototipo en lenguaje Java. Se ponderan los términos que forman parte de las ontologías a partir de una función peso basada en un corpus de documentos. Luego, se hace uso de un extractor de palabras claves (keywords) para el armado del vector de términos del documento que será catalogado.

Para el proceso de catalogación se compara cada término del vector del documento con

cada término en la ontología y también se considera la frecuencia del término en el documento a catalogar.

A partir de que no fue posible disponer de una ontología de dicho dominio en idioma español abierta, surge la necesidad de diseñar esta ontología, por lo cual, en este trabajo se propone el diseño y la construcción de una ontología en español de algunas áreas de las Ciencias de la Computación. Para el diseño de la ontología se utiliza la metodología Methontology. Para que la ontología resulte completa y correcta fue necesario explorar distintas fuentes confiables y científicas. En nuestro trabajo, la base de conocimiento está enriquecida por sitios web tales como ACM, CiteSeerX y Springer y bibliografía relevante dentro del dominio de interés.

El resto del trabajo se estructura de la forma siguiente, en la Sección 2 se analizan algunos extractores de palabras claves actuales y se muestra una comparación de los mismos; en la Sección 3 se presentan conceptos básicos de ontologías, el estado del arte de las diferentes metodologías para el diseño de una ontología y una comparación de ellas; la Sección 4 refiere al diseño de la ontología que se utilizará aplicando Methontology; en la Sección 5 se propone una arquitectura para la catalogación y se describe paso a paso la propuesta; en la Sección 6 se presenta la experimentación y los resultados obtenidos, utilizando precisión para evaluarlos; por último en la Sección 7 se realiza una conclusión del trabajo y los posibles trabajos futuros que se abren.

# Capítulo 2

# Extractores de información

Algunos ejemplos que pueden servir de disparadores para entender la problemática que intenta solucionar la Extracción de información (EI) pueden ser, una empresa que necesita saber el impacto de un nuevo producto en el mercado, el gobierno está recolectando información de un desastre natural ocurrido y quiere informar los últimos datos recolectados a los servicios de urgencias, un grupo de médicos está investigando un nuevo tratamiento y quieren conocer todas las formas posibles en que un grupo de proteínas pueden interactuar con otras proteínas y cuáles serían los resultados exactos de esta interacción.
En una mirada más general podemos clasificar las necesidades que surgen de una forma más abstracta, se realiza una solicitud de información, la respuesta a tal solicitud se puede presentar de forma no estructurada, en texto o imágenes. El usuario no puede procesar semejante cantidad de información mientras que a una computadora se le dificulta retornar de manera precisa y de la forma que el usuario desea, ya que la mayoría de los recursos de donde se extrae la información se encuentran no estructurados, es aquí donde aparece Extracción de información para presentar soluciones a las diferentes problemáticas.
Como podemos ver en [6], la definición de Extracción de información fue cambiando con el pasar de los años. A finales de los noventa Riloff y Lorenzen's definen:

> *"Los sistemas EI, obtienen información de un dominio específico de texto redactado en lenguaje natural. El dominio y el tipo de información que se extrae se deben definir con antelación. Los sistemas de EI a menudo se centran en la identificación de objetos, tales como referencias a personas, lugares, empresas, y los objetos físicos. [...] Patrones de extracción de un dominio específico (o algo similar) se utilizan para identificar la información relevante."*

El problema que en la actualidad podemos observar en esta definición es que está limitada sólo a un dominio específico, hoy existen muchos extractores de información

sin necesariamente tener que indicarle con anticipación el dominio en el cual buscará esa información. Luego en [6] podemos observar una definición más adecuada:

> *"EI es la identificación y clasificación consecuente o concurrente y estructuración en clases semánticas, de información específica que se encuentran en las fuentes de datos no estructurados, como texto de lenguaje natural, haciendo que la información sea más adecuada para las tareas de procesamiento de la misma."*

En el siguiente punto vemos la definición de extractor de palabra claves que es un caso particular de EI.

## 2.1. Extractor de palabras claves

Un extractor de palabras claves (PC) o *keyword extractor* se encarga de obtener un conjunto de términos a partir de un texto que representen al texto coherentemente, extrayendo del mismo los términos más importantes del texto, lo deseable es que sea un conjunto pequeño de términos y frases (ver más información en [21]). Cuando observamos un documento científico por lo general se identifica al principio del mismo una etiqueta llamada "Palabras claves" o "Keywords" que también forman parte de los metadatos del documento, por lo general estas frases y PC son seleccionadas por el autor.
Existen muchas herramientas de extracción de PC, la mayoría de estas no sólo extrae PC sino que también realizan otros tipos de análisis relacionados al procesamiento de texto y recuperación de información, como extracción de PC a partir de un vocabulario controlado, índices de temas o tópicos, etiquetado automático y algunas más. Podemos nombrar las siguientes aplicaciones de extracción TexLexAn [24], Alchemy [15], KEA [7], Semantria [17] y varias más. Las cuatro nombradas se describen a continuación.

### 2.1.1. Keyword Extraction Algorithm (KEA)

KEA es una herramienta diseñada para asignar palabras claves y frases claves (FC) que sean representativas del contenido de un documento, esta tarea es llamada indexación de frases claves (*keyphrase indexing*). Por ejemplo en las publicaciones académicas por lo general están acompañadas de una parte que dice *Keywords* donde se encuentran estos términos claves y pueden ser utilizados en un futuro para otros propósitos.
Por defecto se procesan los documentos en idioma Inglés pero esto se puede cambiar si se reemplaza o edita el archivo que contiene las Stopwords, modificándolo por las *Stopwords* (es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc., también llamadas "Palabras Vacías") del idioma que se necesite y también cambiando el Stemmer (algoritmo que reduce una palabra a su forma raíz,

stem o lema).
También en [7] se puede descargar diferentes tesauros en formato SKOS y configurarse como vocabulario controlado. Actualmente se encuentran el tesauro Agricultural Agrovoc, Medical Subject Headings (MeSH), vocabulario High Energy Physics (HEP) y tesauro Alcohol And Drugs (AOD Thesaurus).
Para la extracción de las posibles PC, KEA extrae n-gramas (subsecuencia de n elementos de una secuencia) de una longitud predefinida que no empiecen ni terminen con *stopword*, que coincidan con términos que se encuentran en el tesauro, además si el tesauro define relaciones entre términos que no son descriptores y términos que sí lo son, reemplaza cada descriptor por un no-descriptor equivalente. Vale decir que para cada candidato se calcula TF-IDF (*Term frequency - Inverse document frequency*), el porcentaje de la primera ocurrencia del término en los documentos, ya que los términos que tienden a aparecer en las primeras páginas son mejores candidatos, cantidad de palabras que componen la frase y el grado del nodo que representa a la frase. Esto implica contar las frases que están semánticamente relacionadas con el nodo de la frase candidata, para esto se requiere consultar las relaciones del tesauro. Al final del proceso para cada documento se crea un archivo ".key" con el mismo nombre del documento donde están contenidas las FC del documento. En la Figura 2.1 se observa el procesamiento del documento llevado a cabo por KEA:

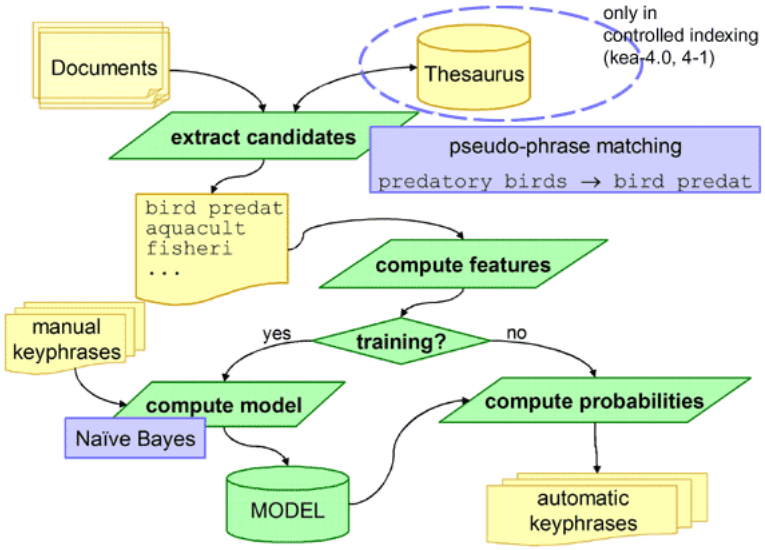



Figura 2.1: Arquitectura de KEA, extraída de [7]

### 2.1.2. Semantria

Semantria es una API que se utiliza desde Microsoft Excel donde su función principal es el análisis de sentimientos [17], extracción de entidades y palabras claves. Soporta

los idiomas, Inglés, Español, Francés, Portugués, Alemán, Mandarín, Italiano, Coreano, Japonés y algunos más. Las partes fundamentales de la arquitectura de Semantria se encuentran en la nube (cloud). A continuación se ven brevemente cada funcionalidad que ofrece.

**Análisis de sentimientos:**

El análisis de sentimientos es el proceso de detectar tres categorías de sentimientos que son positivos, negativos o neutrales en un texto. Estos análisis surgen ante la imposibilidad de leer miles de documentos para un humano y poder sacar conclusiones que faciliten un objetivo en particular. Semantria se basa en el procesamiento de lenguaje natural (PLN) y facilita soluciones más consistentes y más rápidas de las que un humano pueda obtener. El proceso es el siguiente:

1. Se parte el documento en partes básicas llamadas etiquetas POS (POS tags) las cuales representan los elementos estructurales de una sentencia (adjetivos, pronombres, verbos, etc.).
2. Aplica algoritmos (no detallado en [17]) para identificar frases de sentimientos.
3. A cada uno de esos sentimientos se le asigna un peso (score) entre -10 y 10.
4. Luego se combinan todos los pesos para determinar los sentimientos del documento en general. Es un valor entre -2 y 2.

**Concept Matrix y Deep Matrix:**

Concept Matrix es una parte de Semantria que se encarga de consultar a Wikipedia para entender y medir la distancia semántica que existe entre las palabras. También usando algoritmos de Deep Learning con Concept Matrix es posible que reconocer las palabra y frases de una manera más humana.

Categorización

Aquí se utiliza Concept Matrix que se encarga de recorrer Wikipedia para armar relaciones entre las palabras y los conceptos. Luego se necesitan algunas palabras de ejemplos para que se pueda terminar la categorización entera.

Extracción de entidades nombradas

Automáticamente se extraen nombre propios del texto y se determinan los sentimientos asociados al documento. Entidades como personas, lugares, empresas y marcas son clasificadas, etiquetadas y asignadas a un score (peso). Existen algunas entidades preinstaladas en Semantria y también se puede agregar una lista personalizada.
Cada entidad nombrada tiene asociado los siguientes parámetros:

- Entidad: diferentes formas de expresar la misma entidad se asocian a una única forma de nombrarla, por ejemplo "El Zorzal Criollo" y "Zorzalito" se asocian a "Carlos Gardel".
- Sentimiento: puede ser positivo, negativo o neutral para cada mención de la entidad nombrada.
- Evidencia: El número de frases de sentimiento que sean asociadas a una determinada entidad. El score (peso) varía entre 1 y 7.
- Confidencia: Un valor numérico de 0 a 1 que indica si una entidad coincide con su búsqueda booleana.

Consultas

Las consultas están implementadas con funciones booleanas de búsqueda. El motor de búsqueda es literal, no como las Categorías que se basan en Concept Matrix, las consultas se constituyen usando operadores booleanos y lógica. Cualquier término que no esté explícitamente en la consulta es descartado.

Extracción de tema del documento

Los temas de los documentos son sintagmas nominales extraídos del texto que pueden ser utilizados para identificar las principales ideas del contenido del documento. Se asigna un peso o score a cada tema extraído para poder discernir si un tema es más relevante que otro y además el algoritmo tiene en cuenta el contexto y la posición en el texto del sintagma nominal para la asignación de pesos.

Resumen automático

Semantria resume una base del texto a nivel oracional, lo cual significa que se seleccionarán las sentencias más representativas al contexto del documento. En conjunto estas sentencias forman una sinopsis concisa del texto original. El motor (engine) establece una asociación conceptual entre los sustantivos relacionados, incluso cuando los conceptos están ubicados en diferentes partes del documento.

### 2.1.3. Alchemy API

Alchemy API es un conjunto de soluciones basadas en procesamiento de texto y procesamiento de imágenes. Las herramientas que provee Alchemy API están orientadas a grandes empresas y son principalmente para hacer minería de datos en texto. Sólo describiremos brevemente las características más importantes que establece un punto de contacto con esta Tesina. En la Figura 2.2 se puede observar cómo trabaja:

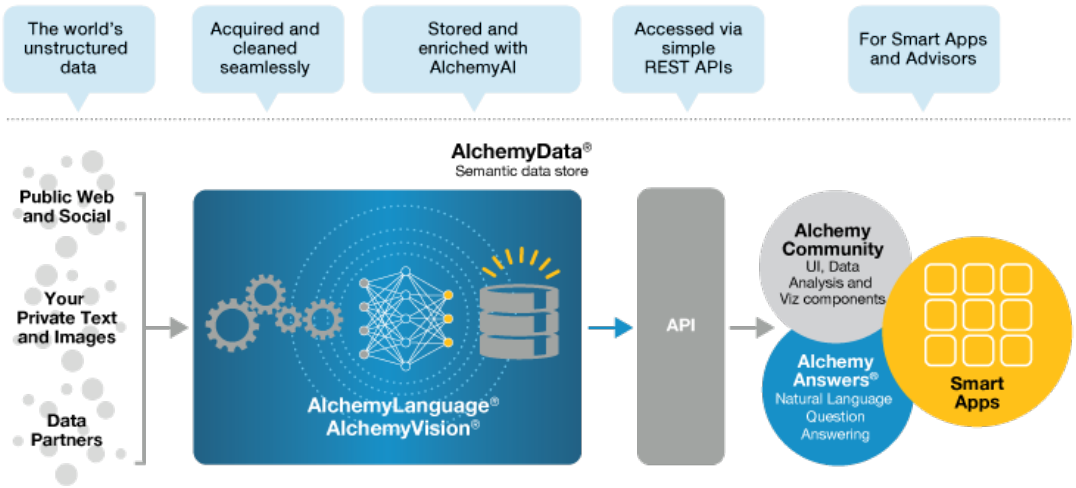



Figura 2.2: Arquitectura de Alchemy API, extraída de [15]

Dentro del conjunto de herramientas de Alchemy API se tiene, Extracción de entidades (Entity extraction), Análisis de sentimientos, Etiquetado de conceptos (Concept Tagging), Extracción de relaciones, Extracción de palabras claves, Taxonomías, Extracción de autores, Detección de idiomas, Extracción de texto, Detección de Feed entre otros. Como en la parte anterior referida a Semantria vimos la descripción de algunas de estas funcionalidades, sólo la funcionalidad y no la implementación, a continuación describiremos brevemente la extracción de PC que realiza Alchemy.

**Extracción de palabras claves**

Se encuentran las palabras claves (PC) del texto y se construye un ranking con las mismas, se pueden asignar sentimientos a cada PC. La extracción de PC puede realizarse sobre URLs, documentos HTML y texto plano, el idioma se detecta automáticamente para luego efectuar el análisis. El algoritmo utilizado para generar el ranking de PC no se encuentra expuesto en el sitio web de Alchemy debido a que es de licencia privativa.

### 2.1.4. TexLexAn

TexLexAn [24] está compuesto por:

**La interfaz gráfica**

Provee las opciones más usadas, es limitada con respecto a la linea de comandos, permite drag'n drop de archivos local y links de Internet, crear nuevas clases. Requiere las aplicaciones *get*, *antiword*, *pdftotext*, *ppthtml*, *odt2txt*, *TexLexAn* and *lazylearner*.

**Motor analizador-clasificador-resumidor**

Realiza el trabajo principal, reconoce los formatos UTF-8 y ISO-8859-1 en forma texto o html, reconoce lenguaje y luego los convierte a ASCII. Luego el texto es tokenizado y se extraen las PC.

- El clasificador:

  las PC se buscan en un diccionario de categorías o clases, cuando una PC es encontrada se almacena la categoría del texto y se le asigna un valor a esa categoría.

- El resumidor:

  se encarga de extraer las sentencias más importantes. TexLexAn usa tres tipos de métodos para resumir. El primero se basa exclusivamente en PC que se extraen del texto. El segundo método se basa en el diccionario de categorías, la clasificación del texto provee una lista de palabras que pertenecen a determinadas categorías, estas palabras son buscadas en cada sentencia del texto y permiten establecer un valor o peso para cada sentencia, luego la sentencia más relevante es la de mayor peso y es la que se incluye en el resumen. El tercer método está basado en una lista de *cues expressions*. Las sentencias de texto que tienen *cues expressions* se extraen para ser incluidas en el resumen. Se pueden combinar el primer método y el segundo método o el primer método y el tercer método para obtener un resumen más corto. Esta decisión de combinar la lleva a cabo una rutina llamada "smartdecision" que intenta encontrar las mejores coincidencias de algunas características del texto actual con textos anteriores, además de tener en cuenta los resúmenes generados previamente para luego decidir como combinar los métodos de resumen.

- El análisis de sentimientos:

  opera similar al clasificador, pero utiliza una base de conocimientos estática y algunas reglas básicas de sintaxis para evaluar el sentido que expresa cada sentencia. El sentido de la sentencia se identifica en forma bipolar como 'bueno' o 'malo'. Se puede elegir palabras o expresiones para que el análisis de la sentencia se limite a la contención o no de las mismas.

- El motor de aprendizaje:

  basado en el algoritmo *Perceptrón* para actualizar las clases del diccionario como nuevas clases, nuevas PC o actualizar el peso de las PC existentes. Luego de ejecutar el *analizador- clasificador-resumidor*, se ejecuta el motor de aprendizaje, por lo tanto los diccionarios son actualizados.

- Programas secundarios:

  *sis* y *tlsearch* son utilizados para encontrar términos dentro de los resúmenes archivados para recuperar los documentos originales.

De las herramientas analizadas he seleccionado TexLexAn, ya que permite:

- Listar palabras claves.
- Hacer un conteo de repetición y estimación del promedio de palabras.
- Estimar el tiempo de lectura y la dificultad del texto.
- Categorizar el texto.
- Realizar un resumen por extracción.
- Efectuar la búsqueda de plagiarismo.
- Evaluar sentimientos.

El soporte de formatos de texto es bastante amplio. Entre ellos podemos encontrar: links url, archivos html, pdf, odt, ppt, doc. También posee una GUI para facilitar su uso.

## 2.2. Comparación y selección de extractores de palabras claves

Hasta aquí hemos visto las características y herramientas que nos proveen algunas empresas o proyectos de código abierto. En principio podemos agrupar por un lado a Semantria y Alchemy API que son empresas que se dedican a vender sus productos de análisis de texto, la gama de herramientas va desde extractores de palabras claves hasta análisis de big data tomando como datos de entrada, según lo que ofrecen en sus sitios web, cualquier tipo de dato que tenga la empresa, datos no estructurados, documentos en cualquier formato, páginas webs y más. Entre sus principales características podemos nombrar, análisis de sentimientos, detección automática del lenguaje, extracción de entidades, construcción automática de taxonomías, resumen de texto. Sólo le queda al cliente analizar los datos que proveen estas herramientas para cumplir su objetivo. Además soportan los idiomas más populares como Inglés, Francés, Portugués, Español, Alemán, Japonés y muchos más. La ventaja que tiene Alchemy API sobre Semantria es que al menos deja efectuar 1000 transacciones por día gratis mientras que en Semantria todo es pago, aunque esto comparado con KEA y TexLexAn es una desventaja ya que no tienen límites de ningún tipo al ser de código abierto.
Volviendo a lo que es una transacción vale decir que, no es lo mismo una transacción en Alchemy API que en Semantria. Aquí sí Semantria lleva la ventaja ya que una transacción es igual a consultar por un documento o un tweet por ejemplo. Mientras que en Alchemy API podemos llegar a necesitar más de una transacción por documento, porque para cada documento hay un límite de 50 kilo bytes y para html 600 kilo bytes.

Sólo en el texto plano del documento o del html.
Por otro lado tenemos KEA y Texlexan que a diferencia de los dos anteriores son de código abierto. TexLexAn es la herramienta elegida para este proyecto de tesis. No está demás decir que podemos modificar el código, como modificar los algoritmos implementados, además que en ambos podemos cambiar el vocabulario controlado por el que querramos, en el caso de KEA está en formato SKOS mientras que en TexLexAn hay que modificar el archivo que funciona como diccionario, también podemos modificar agregando términos a los ya existentes en la web o los que vienen con ambos proyectos. KEA sólo soporta Inglés pero si se provee otro stemmer y diccionario de stopwords del idioma de interés es posible utilizarlo en el texto del mismo idioma. En este punto Texlexan tiene ventaja sobre KEA ya que soporta Inglés, Francés, Alemán, Español e Italiano. Otra ventaja importante es que acepta como entrada texto plano (txt), documentos (pdf, odt, ppt y doc) y urls mientras que KEA sólo acepta txt, esto se debe a que TexLexAn requiere los paquetes *pdftotext*, *odt2txt*, *ppthtml* and *antiword* para operar sobre los textos. En un marco general, TexLexAn ofrece muchas menos herramientas que Semantria y Alchemy API, TexLexAn permite análisis de sentimientos, resumen automático, extracción de PC, estimación de lectura de texto, categorización de texto, entre otras, que es más que suficiente para este proyecto de tesina, mientras que KEA sólo permite extracción de PC e indización de las mismas, que también es una opción pero el problema es que no soporta idioma Español. Otra ventaja en sentido utilitarista es que TexLexAn lo podemos compilar (Java) o también podemos descargar de [23] donde existe una versión compilada que funciona muy bien en Ubuntu 9.04, mientras que en KEA no está la opción de descargarlo compilado. En la tabla siguiente se presenta un cuadro comparativo de estas cuatro herramientas para el análisis de nuestros textos.

| Característica | Aplicación | | | |
|---|---|---|---|---|
| | **Semantria** | **Alchemy API** | **TexLexAn** | **KEA** |
| **Extraer PC** | Sí | Sí | Sí | Sí |
| **Entrada** | Excel, documentos. | páginas web, imagenes, emails. | páginas web, documentos, presentaciones. | Texto |
| **Salida** | Gráfico torta, gráfico barra, texto. | Datos procesados en aplicación, xml, json, rdf. | Texto | Texto |
| **Licencia** | Privativo, soft. propietario. | Privativo, sof. propietario. | Código abierto | Código abierto |
| **Límite consulta x día** | Según aplicación 3334, 33334, 166666, pago. | 1000 gratis, >90000 pago. | Sin limites, gratis. | Sin limites, gratis. |
| **Idioma del documento ingresado** | Inglés, francés, portugués, español, alemán, mandarín, italiano, coreano, japonés. | Inglés, francés, portugués, español, alemán, mandarín, italiano, coreano, japonés. | Inglés, francés, español, alemá, italiano. | Inglés, pesonalizable a otros. |
| **Extracción de entidades** | Sí. | Sí. | Sí. | Sí. |
| **Distingue sinónimos y duplicados** | Sí. | Sí. | Sí. | Sí. |
| **SO/Plataforma** | Windows, SDK multiplataforma. | App MultiSO, SDK multiplataforma. | Linux, SDK Java. | App MultiSO. |

Cuadro 2.1: Comparación de los extractores analizados

# Capítulo 3

# Ontologías

### Introducción

Ante la enorme cantidad de texto no estructurado en la web y el veloz crecimiento del mismo las búsquedas y clasificaciones de temas de interés para determinados sectores de la sociedad que se corresponden con dominios en particular, resulta una tarea tediosa. Además existe un gran cantidad de textos no supervisados por expertos, en dominios donde es fundamental la rigurosidad científica para realizar búsquedas específicas. Quizás resulte fácil encontrar una descripción general de un tema determinado, pero cuando es necesario ahondar en determinados temas, que aquí llamaremos categorías, es decir no sólo encontrarlos sino que lo que encontrado sea lo deseado, es donde surge la necesidad de crear, si no están creadas al momento, categorías que tengan un rol descriptivo en cuanto al contenido de un texto. Estas categorías las denominamos Ontologías de dominio. Es de suma importancia y debe quedar en claro la diferencia que la Ontología de dominio es una base de conocimiento que se construye con expertos en el dominio y que también permite realizar razonamientos realizados por sistemas inteligentes sobre dicha estructura, a diferencia de una base de datos que sólo posee ciertas funciones primitivas para obtener la información deseada.
Otra ventaja que tienen las ontologías es la posibilidad de reutilizar y compartir una base de conocimiento, este aspecto se puede observar en detalle en [20]. Por ejemplo si se diseña una Ontología de vinos, donde se pueden tener categorías como tinto, blanco, malbec, chablis, aroma, frutos rojos, etc., esta ontología se puede utilizar para clasificar documentos de vinos y también se puede utilizar para realizar ciertos razonamientos para acompañar una comida en particular, donde la base de conocimiento sobre las comidas o platos pueden formar parte de la misma ontología de los vinos o en una ontología aparte que represente el dominio de comidas o platos.
A continuación se detalla la definición de Ontología en general, existen varias definiciones que si bien comparten similitudes en cuanto a lo teórico, cada una adopta cierta particularidad cuando cambia el contexto donde se utiliza. Luego se describen metodologías de diseño de ontologías y se comparan dependiendo de sus características. Se

selecciona la metodología Methontology para implementarla sobre las áreas Aprendizaje Automático, Agentes Inteligentes y Teoría de Bases de Datos, dentro del dominio de las Ciencias de la Computación.

**Definición**

Antes de definir los elementos que componen una Ontología, repasemos algunas de las primeras ideas y definiciones que surgieron primeramente en la Filosofía, más precisamente en la Metafísica. Ontología es el "estudio de lo que hay", de la existencia y no existencia de cosas y cómo estas cosas se relacionan y no se relacionan, entre sí. El primero que se refirió a la idea fue Aristóteles llamándola Filosofía Primera, luego sus discípulos la llamaron metafísica, entre el 500 y 600 A.C. Luego en el siglo XVII, Leibniz escribe:

> *"Ontología es la ciencia de algo y de nada, del ser y del no ser, de la cosa y del modo de la cosa, de la sustancia y el accidente'.'*

Otros pensadores han trabajado exhaustivamente sobre Ontologías como Rudolph Gockel, antes de Leibniz, Jean Le Clerc y Christian Wolff.
Volviendo al campo de la Inteligencia Artificial, cuando nos referimos a Ontologías el término mantiene en cierta parte el significado filosófico pero estamos situados en un campo puramente científico donde es necesario hacer pie en definiciones concretas, aplicables. En este sentido, citamos la definición de Nicola Guarino [10], que resulta más adecuada para esta Tesina en particular,

> *"... en su uso más típico en IA, una ontología es un artefacto ingenieril constituido por un vocabulario específico para describir una cierta realidad, más un conjunto de supuestos explícitos concernientes al significado pretendido de las palabras del vocabulario. Este conjunto de supuestos tiene generalmente la forma de teorías lógicas de primer orden, donde las palabras del vocabulario aparecen como predicados unarios o binarios, respectivamente llamados conceptos y relaciones. En el caso más simple, una ontología describe una jerarquía de conceptos relacionados por relaciones de subsunción; en los casos más sofisticados, se añaden axiomas para expresar otras relaciones entre conceptos y restringir la posible interpretación."*

Existen varios autores que han definido Ontología, como Tom Gruber [12], Willem N. Borst [5], John F. Sowa [27], entre otros. Pero entendemos esta definición como las más acertada para este trabajo.
El tipo de Ontología sobre el que se va a trabajar es sobre Ontologías de Dominio. Este tipo contiene los conceptos asociados a un dominio en particular con sus subcategorías que lo componen, posee descripciones sobre la estructura y contenido del conocimiento del dominio. Los elementos que componen una ontología son:

Clases o conceptos:

Definen un orden jerárquico para los conceptos que componen la ontología. Cuando decimos concepto nos referimos a una idea general del término, un concepto puede ser una funcionalidad, un proceso, una estrategia.

Atributos:

Están contenidos en los conceptos, le dan forma a los mismos formando parte de su estructura interna, pueden ser heredados por las instancias del concepto en relación a la taxonomía definida o específicos donde aparecen únicamente en el concepto.

Instancias:

Es un caso concreto de un concepto, donde todos los atributos del mismo toman un valor determinado.

Relaciones:

Principalmente se definen para representar algún tipo de interacción entre los conceptos, le dan forma a la taxonomía, algunos ejemplos más comunes de relaciones son subclase-de, parte-de, parte-exhaustiva-de.

Funciones:

Indican el cálculo de algo necesario para algún concepto, es un caso particular de relación, por ejemplo asignar-fecha, precio-de.

Axiomas:

Agregan expresividad a la estructura ontológica, permiten escribir reglas que se verifican en las clases y sus atributos para enriquecer ciertos razonamientos que se realicen sobre la ontología. A las ontologías que incluyen axiomas las llamamos Ontologías Pesadas mientras que a las que no las llamamos Ontologías Ligeras.

## 3.1. Clasificación de ontologías

Existen varios enfoques desde donde se pueden clasificar las ontologías en distintos grupos, en [8] y [3] se puede observar la gran cantidad de clasificaciones existentes hasta el momento. Se ha elegido la clasificación que se divide en los siguientes grupos:

Ontologías para la representación de conocimiento:

Modelan las conceptualizaciones que subyacen en los formalismos de representación de conocimiento. En general están enfocadas a un uso particular del conocimiento que describen.

Ontologías genéricas:

Definen conceptos considerados genéricos en diferentes áreas. Son reutilizables en diferentes dominios. Se llaman también ontologías abstractas o superteorías porque permiten definir conceptos abstractos, y dichas ontologías pueden ser usadas para definir conceptos de forma más específica en diferentes dominios. Como ejemplos podemos ver la taxonomía, la mereología, la topología y la teoría general de sistemas.

Ontologías de dominio:

Definen conceptualizaciones específicas del dominio. Las metodologías actuales de adquisición de conocimiento distinguen entre ontologías y conocimiento del dominio, porque el último describe situaciones factuales del dominio, mientras que las ontologías imponen descripciones sobre la estructura y contenido del conocimiento del dominio.

Ontologías de aplicación:

Están ligadas al desarrollo de una aplicación concreta. Tales ontologías cubren los aspectos relacionados con aplicaciones particulares. Típicamente, estas ontologías toman conceptos de ontologías del dominio y genéricas, así como métodos específicos para realizar la tarea, por lo que no son muy adecuadas para ser reutilizadas.
Esta Tesina se propone utilizar Ontologias de dominio, para conceptualizar algunas subáreas, Aprendizaje Automático, Teoría de Bases de Datos y Agentes Inteligentes del dominio de Ciencias de la Computación. Son Ontologías de dominio porque los términos (instancias) que la componen están familiarizadas entre sí y en conjunto pertenecen al dominio que se está representando. Cuando decimos "pertenecen al dominio" nos referimos a que si buscamos en la web algún sitio, documento o foro científico que esté enmarcado por ejemplo en el dominio de Machine Learning, probablemente los términos que aparecen en la Ontología aquí definida aparezcan también en dicha fuente o discusión. Esto nos refiere que hay una contextualización de los términos claves que aparecen en las distintas fuentes.
Además de esto, la forma en que se jerarquizan los términos en la estructura de la Ontología por medio de las relaciones definen no sólo el orden de los términos sino también el sentido que adoptan los mismos en determinado dominio.

## 3.2. Metodologías para diseñar una ontología

En el momento que nos proponemos definir una ontología es importante adoptar alguna de las metodologías existentes para construir una ontología. La serie de pasos

que componen una metodología nos puede ayudar a que el proceso de construcción no sea tan caótico. Se puede observar una comparación de las diferentes metodologías en [1] y en [18].
Actualmente existen muchas metodologías de diseño de ontologías. Sin embargo cada una tiene su particularidad y está orientada a ciertos tipos de contextos y dominios, no sólo en la definición de clases, atributos, instancias y relaciones sino en la forma de estudiar sus aplicaciones y el trabajo previo en el dominio dado. A continuación repasaremos de forma breve algunas metodologías para terminar luego en una descripción más amplia de Methontology que es la seleccionada para este trabajo. Las metodologías' "Cyc", "Grüninger y Fox" y "Uschold y King" se desarrollaron entre 1990 y 1995 mientras que las otras a partir de 2002.

**Metodología Cyc**

Comenzó como una parte de un proyecto para representar el comportamiento humano con una ontología [16]. Las etapas que luego llevaron a la creación de una ontología fueron organizadas y asi surgió esta metodología. Básicamente posee dos etapas, primero la búsqueda manual de conocimiento de diferentes fuentes, y como segunda etapa la utilización de herramientas de procesamiento de lenguaje natural para procesar la información obtenida de las fuentes y/o técnicas de aprendizaje automático para seguir enriqueciendo la ontología. La ontologías se escribe en un lenguaje llamado CycL, y se inserta el conocimiento en forma de aserciones.

**Metodología de Grüninger y Fox**

M. Grüninger y M. S. Fox comenzaron a detectar varios problemas de modelado de conocimiento en empresas industriales. Entonces empiezan a formularse preguntas en lenguaje natural que la ontología a construir debería contestar [13], la potencialidad de la ontología de poder o no contestar esas preguntas la llaman "competencia de la ontología". Esta etapa es la primera de este método. Luego la segunda etapa consiste en definir la terminología que formará parte de los objetos, atributos y relaciones de la ontología. El tercer paso consiste en cuando sea posible, especificar las definiciones y restricciones de la terminología. En su momento Grüninger y Fox especificaron con lógica de primer orden implementada en Prolog. También testearon la competencia de la ontología probando teoremas de completitud.

**Metodología de Uschold y King**

En [26] y [28] podemos observar que está compuesta por las siguientes cuatro etapas.

1. Identificar la propuesta: Es decir para qué se está creado la ontología y cuál es el uso que se va a hacer de la misma pensado en el espectro de usuarios que la utilizarán.

2. Contrucción de la ontología:

   - *Captura:*
     - Identificar los conceptos claves y las relaciones en el dominio de interés.
     - Desambiguación a nivel textual de los conceptos y relaciones definidos anteriormente.
     - Identificación de términos para referenciar a los conceptos y relaciones.
     - Estar de acuerdo con todo lo hecho hasta aquí.

   - *Codificación:* Este paso consiste en la representación explícita de la conceptualización capturada en la subetapa anterior en algún lenguaje formal; por lo tanto se codifica en ese lenguaje.

   - *Integrar con otras ontologías:* durante la captura y codificación, nos puede surgir la pregunta de cómo y cuándo utilizar las existentes metodologías. Es un problema difícil de resolver. Skuce [26], postula que si la ontología se va a compartir en grandes comunidades es más trabajoso definir una ontología donde cada individuo de la comunidad esté de acuerdo con ella, entonces sugiere hacer explícita todas las asumpciones que no están tan explícitas en la ontología.

3. Evaluación: Esta etapa es bastante pobre metodológicamente. Lo que se propone es mirar cómo se evalúan las ontologías del mismo dominio y se sostiene que en el mundo real surgirán las mejores evaluaciones de la misma.

4. Documentación: Siempre es deseable que la ontología este documentada, en el sentido de compartirla en otros ámbitos al previamente pensado es fundamental, ya que permite en primera instancia su compresión, y luego su reutilización.

**Metodología TERMINAE**

Es usada para gestionar ontologías ya que es más que una metodología. La herramienta está desarrollada en Java; originariamente pensada para integrar herramientas de ingeniería de conocimiento e ingeniería de la lingüística. Esta última parte permite definir terminologías y relaciones entre los términos como "es-sinónimo-de", entre otras. La parte de ingeniería provee un editor y un navegador para las ontologías. Si la clasificamos en etapas, como se puede observar en [4], se puede describir como primer etapa la

definición de términos que componen un corpus, es decir, hacer una lista de términos. Luego se utiliza LEXTER que recolecta una cantidad determinada de términos para luego ser seleccionados por un experto. Como tercera etapa se listan todas las nociones o significados de cada término y se traslada esta a algún formalismo. Finalmente se analiza si se inserta o no en la ontologías dependiendo de la validez de la inserción.

**Metodología Ontology Development 101**

Está compuesta por siete etapas simples y rápidas de aplicar, no agrega algo nuevo a los que son todas las metodologías en su conjunto, sin embargo una de sus etapas es investigar ontologías existentes para poder reutilizarla. Por más que esto resulte trivial a veces no es tenido en cuenta por los diseñadores. Las etapas se obtuvieron de [22] y son las siguientes:

1. Determinar el dominio y el alcance de la ontologías: Para determinar esto se proponen cuatro preguntas básicas:
   - ¿Cuál es el dominio que cubrirá la ontología?
   - ¿Para qué se usará la ontología?
   - ¿Para qué tipo de preguntas la ontología proveerá respuestas?
   - ¿Quién usará y mantendrá la ontología?

   Vale decir que la motivación que lleva a plantear preguntas es la metodología de Grüninger y Fox.
2. Considerar el reuso de las ontologías existentes: Se debería chequear si ya existen ontologías del dominio que se quiere definir. El reuso de las ontologías existentes puede llegar a ser un requerimiento si la aplicación necesita interactuar con otras que actualmente ya utilizan un vocabulario controlado.
3. Enumerar los términos importantes de la ontología: Es importante escribir una lista de todos los términos que se consideran relevantes , ¿cuáles son los términos que nos interesa nombrar?, ¿qué propiedades tienen estos términos?, ¿qué nos gustaría decir acerca de ellos?
4. Definir las clases y la jerarquía de cada clase: Las siguientes técnicas de cómo definir las jerarquías tienen base en la metodología de Uschold y Grüninger:
   - La técnica *Top-down* comienza definiendo los conceptos más generales del dominio y luego las especializaciones subsecuentes de cada concepto.
   - La técnica *Bottom-up* comienza definiendo las clases más específicas, las hojas de la jerarquía general y luego se va agrupando en conceptos más generales.

- La última técnica es la *Combinación* de las dos anteriores. Se definen los términos más sobresalientes primero y luego se la generaliza y se la especializa apropiadamente.

Ninguna de las tres técnicas es mejor que otra, todo depende de la vision personal que se tenga sobre el dominio.

5. Definir las propiedades de las clases (slots): Una vez que se han definido las clases, se debe describir la estructura interna de los conceptos. En esta etapa ya se tienen las clases definidas que fueron tomadas de la lista de términos, los términos restantes posiblemente sean propiedades de las clases. Estas propiedades se denominan slots adjuntadas a cada clase. En general hay algunos tipos reconocidos de propiedades que potencialmente son slots, propiedades intrínsecas, propiedades extrínsecas, propiedades abstractas y físicas de un objeto, relaciones con otros objetos.
6. Definir las facetas de los slots: Hay algunas facetas en común en todas las ontologías.
   - Cardinalidad de slot, define cuántos valores puede tener un slot.
   - Tipo de valor del slot, pueden ser:
     - Cadena (string).
     - Un número que puede ser flotante o entero, u otro tipo a tener en cuenta.
     - Un booleano.
     - Valores enumerados, alguno definido por el diseñador.
     - Instancias, permiten la definición de la relaciones entre los individuales (individuals). También debe definirse las clases que acepta la instancia.
   - Rango y dominio de un slot, los valores permitidos para los slots de tipo instancia son llamados Rango. Las clases que son adjuntadas a un slot son llamadas Dominio.
7. Crear instancias: En este último paso se crean instancias individuales de las clases de la jerarquía. Esto requiere, primero elegir la clase, segundo la creación de la instancia individual de esa clase y tercero rellenar los valores del slot.

**Metodología Methontology**

Esta metodología surge de un grupo de investigación del Laboratorio de Inteligencia Artificial de la Universidad de Madrid, sus principales creadores son Mariano Fernández

y Asunción Gómez-Pérez [11].
Se basa fuertemente en la metodología de Uschold y Grüninger, ya que las etapas son similares. Está compuesta por seis etapas que son: especificación, adquisición del conocimiento, conceptualización, integración, implementación y mantenimiento. A continuación se desarrolla cada etapa.

1. Especificación: El objetivo principal de esta etapa es que la ontología quede representada mediante una especificación que puede ser informal, semi-formal o formal utilizando el lenguaje natural. Al menos deben estar especificados:

   - Cuáles son los escenarios posibles y el concreto en los cuales se propone el uso de la ontología, quienes la utilizarían.
   - Nivel de formalidad, según Uschold y Grüninger, altamente informal, semi-informal, semi- formal y rigurosamente formal.
   - Alcance, conjunto de términos que es de interés representar y la granularidad que se intenta.

2. Adquisición del conocimiento: Esta etapa es independiente del proceso de desarrollo concreto de la ontología, ya que está orientada a la investigación de diferentes fuentes que pueden aportar a la idea de ontología que se quiere construir desde un principio. No existe una forma estática que sea aplicable en todos los casos de adquisición de conocimiento, sin embargo los siguientes pasos dan una idea:

   - Entrevista informal con expertos en el dominio para tener por lo menos un bosquejo de la especificación.
   - Análisis de textos informales y el estudio de conceptos desde alguna bibliografía o anotaciones. Estos son potencialmente las clases en la ontología.
   - Análisis de textos formales, es la profundización del paso anterior también buscando en bases de datos bibliográficas, sitios web del dominio que tengan cierto prestigio en general. Formalizar las clases, atributos y valores de éstos, relaciones.
   - Entrevista formal con expertos, esto supone poner a prueba lo armado hasta el momento por el ojo de expertos en el dominio elegido, la evaluación de las clases, atributos y valores de estos, relaciones.

3. Conceptualización: En esta etapa se realiza un modelo conceptual a partir de la especificación formulada, se construye lo que llamaremos Glosario de Términos (GT), donde se incluyen todos los términos hasta el momento. Cuando decimos términos nos referimos a clases, instancias, verbos y propiedades. La idea del GT es analizar el potencial y la usabilidad de cada término para descartarlo o tenerlo en cuenta.

Luego del armado del GT, se deben agrupar los conceptos o clases por un lado y los verbos por otro. En el caso de los conceptos se estructuran en forma de árbol y en el caso de los verbos (relaciones en nuestra definición) se realiza un diagrama. Hasta aquí se tienen dos conjuntos que van a ser descriptos de la siguiente forma, por un lado el *árbol de conceptos* compuesto por:

- *Diccionario de datos:* Todos los conceptos potencialmente usables, sus significados, atributos, instancias, etc.
- *Tabla de instancias de atributos:* provee información acerca de los atributos y los valores que puede asumir.
- *Tabla de atributos de clase:* Describe el concepto mismo y no sus instancias.
- *Tabla de constantes*: especifica la información que permanece inmutable en toda la base de conocimiento.
- *Tabla de instancias:* define las instancias.
- *Árbol de clasificación de atributos:* muestra gráficamente cómo se relacionan las clases.
  Por otro lado tenemos el diagrama de verbos compuesto por:
- *Diccionario de verbos:* expresa el significado de los verbos.
- *Tabla de condiciones:* expresa el conjunto de condiciones que se deben satisfacer antes y después de la ejecución de una acción.

Finalmente están la Tabla de fórmulas y la Tabla de Reglas que reúnen las fórmulas y las reglas de la base de conocimiento.
A continuación en la figura 3.1 podemos ver condensada la etapa 3:

4. Integración: Uno de los objetivos de esta etapa es investigar otras ontologías existentes que puede ayudar a la construcción de la actual para acelerar el proceso de diseño y evitar realizar conceptualizaciones ya creadas y aceptadas por los expertos del dominio. En este sentido se proponen dos pasos:

   - Inspeccionar las existentes metaontologías y observar cuál se acomoda mejor a la que se intenta construir. La meta es garantizar que las definiciones nuevas o que son reutilizadas estén basadas en el mismo conjunto de términos básicos. En el caso que no exista una ontología apropiada se debería empezar a definir una nueva formalmente.
   - Encontrar cuáles son las fuentes ontológicas que proveen definiciones de términos que semánticamente son coherentes con los términos utilizados en la construcción de la ontología actual y que ésto verdaderamente se cumpla.

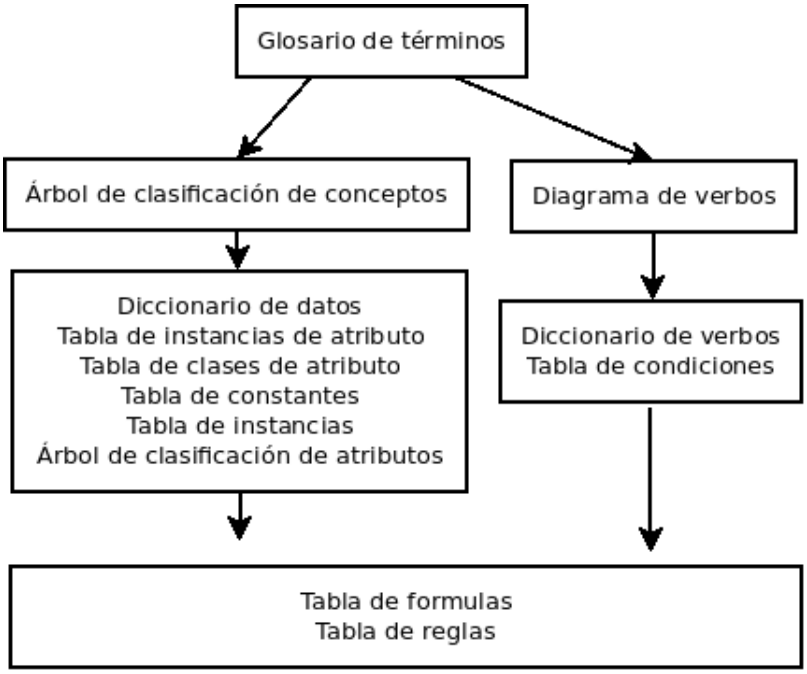



Figura 3.1: Conceptualización de una ontología (extraída de [11])

En el caso que se incluyan términos nuevos, es decir que no formen parte de las ontologías construídas hasta el momento en el mismo dominio, deberá justificarse el por qué de la inclusión de este término con su definición.

5. Implementación: En esta etapa es donde la ontología se implementa en un lenguaje codificado o formal, por ejemplo, Prolog, C++, entre otros. También se puede utilizar herramientas que asisten en la escritura de ontologías como Ontolingua o Protégé. El entorno donde sea especificada la ontología debe al menos contener un analizador sintáctico y léxico para garantizar la ausencia de errores, la posibilidad de exportar la ontología a otros lenguajes para poder reutilizarla en otros entornos, herramientas que permitan que la ontología definida sea flexible como la posibilidad de editarla, agregar y eliminar clases, instancias y relaciones, evaluadores para detectar incompletitud, inconsistencias y conocimiento redundante.

## 3.3. Comparación y selección de las diferentes metodologías

Es difícil comparar algo que no está representado o pensado para comparar como son las metodologías. Es una buena opción sacar las mejores experiencias de los equipos de expertos que diseñaron cada una. Dentro de las etapas de Methontology se puede sintetizar que se encuentran incluídas al menos las etapas más significativas de las otras metodologías, es esa una de las razones por la cual se opta por Methontology. Estas

etapas en común que tienen las demás metodologías con Methontology son:

- Consulta a fuentes de conocimiento sobre el dominio a diseñar.
- Mesurar el alcance y a quiénes está orientada la ontología,
- Capturar los conceptos, relaciones, instancias dentro de un universo de términos.
- Codificación en un lenguaje formal.
- Integración con otras ontologías existentes.

Destacar las ventajas y desventajas de cada metodología en un sentido general y para el caso particular correspondiente a esta tesina es una tarea compleja; en el Cuadro 3.1 se puede leer brevemente algunas de las fortalezas y debilidades de cada metodología. Otro factor que se ha tenido en cuenta es tratar de utilizar completamente la metodología, es decir, intentar explotar cada etapa al máximo y la mayor cantidad de ellas.
Como se describe anteriormente la metodología Cyc utiliza un lenguaje que se llama CycL para definir clases, instancias, relaciones, etc. sobre los términos de una ontología desde cero o ya existente. Pero está más orientada a mezclar con ontologías ya existentes, un conjunto de términos que nos interesen. La desventaja es que no tiene etapas tan desarrolladas en profundidad como Methontolgy por lo que dificulta cómo empezar a aplicar esta metodología. Otra desventaja es que la codificacíon en lenguaje formal está acotada a CycL.
Grüninger & Fox tiene varias desventajas para este caso. En principio que los escenarios que sirven como motivadores son problemas que tienen las empresas y no proyectos de cómo representar un dominio más académico como en este caso, donde la atención está orientada a cómo representar un dominio y no dar respuestas a la industria. Otra desventaja consiste en cómo plantear preguntas que luego la ontología debiera responder. En este caso sólo tenemos una pregunta que es ¿A qué área pertenece este documento? con lo cual se ve minimizada esta etapa de la metodología. Lo que sí es destacable de esta metodología es la traducción a axiomas y teoremas a partir de la formulación de preguntas y la demostración de los mismos. En este sentido es mucho más formal que las demás, aunque innecesario para esta Tesina. Los axiomas son llamados "Teorías de acción" y los teoremas son de completitud y de razonamiento acerca de las acciones definidas como acciones.
Uschold & King no hay nada para criticarle, básicamente tiene todo, el nivel de abstracción necesario y etapas concretas donde es fácil entender lo que se pretende. Podemos decir que indirectamente se utiliza esta metodología ya que Methontology se apoya fuertemente en Uschold & King. Tomando las principales etapas de Uschold & King y especificando con más detalle cada una de ellas, Methontology la mejora agregando

formas de estructurar los diccionarios, clases, atributos y las relaciones que permiten una mayor claridad.
Por otro lado TERMINAE al igual que la metodologia Cyc también se queda en la implementación pura de la ontología. Esto suele suceder cuando a la hora de crear una metodología se piensa sólo en el objetivo final de la misma. Básicamente es un software que obtiene palabras claves (PC) de un corpus que se le provee para luego sacar candidatos de estas PC. Además de esto en este proyecto no se ha seleccionado esta metodología ya que se construye la ontología a partir de PC y no de las principales fuentes donde se nota la experiencia de expertos en el dominio en particular que se esté trabajando. Esto puede llevar a construir una ontología con términos que no son reconocidos o no tienen la mirada de un experto en dicho dominio sino que son términos que aparecen en el corpus que se pasa como parámetro a TERMINAE. A diferencia de esto en este proyecto de tesina se ha consultado a las principales fuentes académicas y no tanto para la construcción de la ontología de dominio, como son libros reconocidos por científicos y sitios webs que se han convertido en referencias de dominio con el tiempo. En este último punto, Methontology sí tiene en cuenta el diálogo con expertos del dominio.
Ontology Development 101 al igual que Uschold y King tiene el nivel justo de abstracción y no deja de ser concreta en cuanto a las tareas que hay que realizar en cada etapa. En la primera además de intentar medir el alcance que tendrá la ontología también se plantean diferentes preguntas como vimos en la metodología de Grüninger y Fox. Otra ventaja importante es que la segunda etapa consiste en la investigación y posible absorción de las ontologías existentes que puedan servir para diseñar la de nuestro interés. Esto es de principal importancia ya que una de las ideas centrales de diseñar una ontología es su posible reutilización en el futuro, en este punto tambien la metodología Methontology tiene en cuenta las existentes ontologías. Volviendo a Ontology Development 101, las etapas siguientes son aplicables a este proyecto ya que consisten en definir los términos de importancia. Mientras que las etapas 5 y 6 no son de utilidad en la ontología que se pretende definir como definir facetas, propiedades de las clases y de las instancias, definir dominios y rangos, entre otras definiciones. Sin embargo es muy buena opción también.
Por último tenemos Methontology que es la seleccionada para aplicar en las áreas que se tienen pensadas diseñar aquí. Las ventajas que se pueden visualizar en esta metodología son varias, en principio que se define desde un principio el grado de formalidad de la ontología y también se rescata de las otras metodologías el planteo del alcance de la ontología y agrega el planteo de pensar cuáles son los usuarios y qué tipo de usuarios tendrán acceso directo o indirecto a la ontología. También propone pensar la propuesta más allá del diseño mismo de la ontología. Esta cuestión es fundamental en este proyecto ya que la ontología sólo sirve como vocabulario controlado para otra meta. En la misma etapa de la especificación deben aclararse las fuentes de conocimiento que se consul-

tan para la construcción de la ontología. Esto es algo primordial si estamos diseñando ontologías de dominio como en este caso, y como consecuencia de esto también está explicitada la etapa de consulta con expertos en el dominio, completando una especie de ciclo donde el conocimento representado en la ontología se refina constantemente. Luego como la mayoría de las ontologías tiene una etapa de definición de las partes de la ontología, clases, instancias relaciones, etc. pero algo también destacable es que antes de esto lo que se plantea es pensar en conceptos que quizás a la hora del diseño no es seguro de que más adelante aparezcan en la misma ontología pero que sin embargo son datos importantes desde el punto de vista de la documentación de la misma. En este sentido, en la publicación original [11] donde se propone Methontology, podemos observar diferentes cuadros y esquemas de cajas que permiten tener una documentación fiable de la ontología, donde la parte de la implementación y la parte del diseño quedan bien distinguidas, algo que en las otras metodologías no queda tan claro. Esto permite que al tener esta documentación pueda ser reutilizable la ontología, sólo observando las etapas aplicadas correctamente podremos compartirlas para que quien quiera pueda implementarla en el lenguaje o framework que le parezca.

| Metodología | Ventajas | Desventajas |
|---|---|---|
| **Cyc** | -Inicialmente hay que consultar fuentes para empezar a diseñar la ontología. | -No se explicitan los pasos a seguir.<br>-Se debe disponer de un sistema de extracción de información para retroalimentar la ontología.<br>-No está claro cómo es la retroalimientación de la ontología. |
| **Grüninger y Fox** | -Se explicitan como una etapa las restricciones de la metodología, en las demás metodologías esto no se tiene en cuenta. | -El contexto de diseño de ontología es comercial y no educativo<br>-El método de planteamiento de preguntas y respuestas para calcular la competencia de la ontología, complejiza en vez de clarificar el diseño. |
| **Uschold y King** | -Los pasos a seguir para el diseño se detallan en mayor profundidad y son claros.<br>-Contiene la especificación formal de la ontología (punto 2).<br>-Está explícita la documentación con el objetivo de reutilización de la ontología. | -El punto de Evaluación, no se explicita exactamente cómo es esa evaluación a realizar. Dejando al mundo real solucionador de dicha evaluación.<br>-No aparace una etapa en donde se listen los términos aunque sean de una forma desordenada. En el sentido de discernir entre concepto, instancia, relación, etc. |
| **TERMINAE** | -Posee una parte de ingeniería lingüistica que permite definir mejor las relaciones entre términos.<br>-Es una herramienta de gestión de ontologías, además de una metodología. | -No esta claro a partir de que fuente se debe empezar a escribir el corpus.<br>-No se especifica el rol del experto que observa y clasifica el corpus (conocimiento de dominio o conocimiento de ontologías). |
| **Ontology Development 101** | -Los pasos a seguir para el diseño se detallan en mayor profundidad y son claros.<br>-Se define la etapa de reutilización de otras ontologías. | -Al igual que Grüninger y Fox la metodología de preguntas y respuestas que no facilita el diseño.<br>-No está detallado en ninguna etapa alguna forma de ordenar el método una vez aplicado de modo tal que sirva de documentación en el futuro. |
| **Methontology** | -Los pasos a seguir para el diseño se detallan en mayor profundidad y son claros.<br>-En la especificación se definen niveles de formalidad y se toman como fuente textos de diferentes niveles de formalidad. | -Es discutible si la etapa de implementación debe estar dentro de la metodología o no.<br>-Puede haber casos donde algunas etapas o partes de etapas no se realicen. Dependiendo del contexto de diseño de la ontología. |

Cuadro 3.1: Algunas ventajas y desventajas de las metodologías'

# Capítulo 4

# Diseño de la ontología propuesta

Como se explicó al inicio del capítulo 3, más allá de qué metodología se ha seleccionado, es importante enmarcarse en alguna para hacer del proceso de construcción y diseño algo más ordenado y poder pensar los objetivos y usos de la ontología en el momento adecuado. Aquí se han diseñado tres áreas de la ontología en el dominio de las Ciencias de la Computación con el objetivo de probar y evaluar resultados de todo el prototipo de clasificación en general, pero también cada ontología en particular es un mínimo aporte al campo de las ontologías en idioma español ya que las existentes en este dominio son escasas y no profundizan tanto en cada subárea de las ciencias de la computación. En la siguiente subsección podemos ver la aplicación de la metodología Methontology para las tres áreas Aprendizaje Automático, Teoría de Bases de Datos y Agentes Inteligentes. Las ontologías propuestas no poseen gran cantidad de clases, instancias y relaciones. Sin embargo es importante disponer de una metodología que funcione como guía más allá del tamaño de la ontología a diseñar, es por eso también que se ha seleccionado una metodología que se escale o que encaje con lo diseñado, básicamente que no sobren etapas en el momento del diseño de las que propone la Methontolgy.

## 4.1. Aplicación de Methontology

### 4.1.1. Especificación

La idea general es diseñar algunas ontologías que se utilicen como vocabulario controlado para buscar las palabras claves del documento dentro de la ontología, particularmente las áreas: Aprendizaje Automático, Agentes Inteligentes y Teoría de Bases de Datos. Partiendo de las principales fuentes que son de referencia para el campo científico de ciencias de la computación.
En un plano más general puede ser utilizado para clasificación de texto y como referencia de las disciplinas que son propuestas, es decir, los conceptos que encierran cada una de las disciplinas, disponer de una base de conocimiento sobre la cual se necesite realizar consultas de términos del área o razonar sobre la misma.

Si observamos cada disciplina como una estructura de árbol, se intenta que cada término tenga una profundidad considerable suficiente para aumentar la especificidad de dicha categoría y no se intenta representar todas las subdisciplinas que se puedan desprender de la raíz, en este caso sería la anchura del árbol. También se tienen en cuenta los términos en idioma Inglés que relacionarán como sinónimos del término principal. A continuación se muestra un documento de especificación como el que se propone en [11].

**Documento de especificación de requerimientos de la ontología**

**Dominios:** Aprendizaje automático, Teoría de Bases de Datos, Agentes Inteligentes.
**Fecha:** 01/04/2015.
**Conceptualizado por:** Fabricio Mahon.
**Implementado por**: Fabricio Mahon.

**Propuesta:** Ontologías de las disciplinas Aprendizaje Automático, Agentes Inteligentes y Teoría de Bases de Datos pertenecientes al dominio de Inteligencia artificial las dos primeras, para utilizarse como base de conocimiento en la clasificación de documento científicos o texto plano.

**Nivel de formalidad:** Semi-formal.

**Alcance:**

- Términos: se investigarán en la adquisición del conocimiento, a priori no son desconocidos.
- Conceptos: partes que componen cada disciplina a nivel conceptual y no como relación que podemos denominar componentes, problemáticas, dificultades, algoritmos, aplicaciones.

**Fuentes de conocimiento:**

- Bibliográficas (Aprendizaje Automático):
  - "Inteligencia Artificial - Un enfoque moderno", Stuart J. Russell y Peter Norvig.
  - "Inteligencia artificial: modelos, técnicas y áreas de aplicación", Maria I. A. Galipienso, Miguel A. C. Quevedo, Otto C. Pardo, Francisco Escolano Ruiz, Miguel A. Lozano Ortega.
- Bibliográficas (Agentes Inteligentes):
  - "Inteligencia Artificial - Un enfoque moderno", de Stuart J. Russell y Peter Norvig.
  - "Intelligent Agent Architectures and Applications", Gautam B. Singh.
- Bibliográficas (Teoría de Base de Datos):
  - "Sistemas de bases de datos", Beynon-Davies Paul.
  - "Fundamentos de Bases de Datos", Silberschatz Abraham, Korth F. Henry y Sudarshan S.
- No bibliográficas: ACM, CiteSeerX, Springer, Wikipedia.

Cuadro 4.1: Especificación de ontologías

### 4.1.2. Adquisición del conocimiento

Al principio se consultó a profesoras de la carrera de grado de Licenciatura en Ciencias de la Computación, específicamente las áreas de Inteligencia Artificial (IA) y Teoría de Base de Datos donde se tomó nota de algunos conceptos generales y también se tomó nota de las posibles fuentes que puedan aportar a la categorización de algunas áreas dentro de IA, para este caso Aprendizaje Automático y Agentes Inteligentes.
Luego se consultaron las fuentes no bibliográficas descriptas en el cuadro de la especificación para tener una idea general de las posibles clases y cómo irían ubicadas jerárquicamente, se realizó un bosquejo general que tiene la forma de tesauro para luego ser evaluado otra vez por las directoras de la Tesina.
Después del segundo encuentro se redefinieron las relaciones de jerarquía y se anotaron algunas posibles relaciones como es-sinónimo-de, es-parte-de y también se evaluó la posibilidad de explotar los conceptos anotados hasta el momento, es decir aumentar la granularidad de estos, explorando las subdisciplinas que aparecen en cada concepto pero además agregando la consulta a las bases de datos bibliográficas emblemáticas en cada dominio, de este modo indirectamente se tiene una visión de los expertos que permite reordenar las jerarquías, agregar términos claves y eliminar términos no tan conocidos o desconocidos.

### 4.1.3. Conceptualización

Construimos un glosario de términos, luego agrupamos en *Conceptos* por un lado y *Verbos* por otro, dentro de la parte de *Conceptos* para este caso en particular hay pasos de la metodología que no se aplican como son los respectivos a constantes y atributos ya que las ontologías aquí definidas no los poseen. A continuación podemos observar el Glosario de términos (GT) y luego las definiciones de tablas y declarativas con respecto a los *Conceptos* y luego las definiciones con respecto a los *Verbos*.

**Glosario de términos (Aprendizaje Automático):** aprendizaje automático, aprendizaje supervisado, programación lógica inductiva, algoritmo de vecinos más cercanos, aprendizaje no supervisado, clustering, aprendizaje semi-supervisado, algoritmos genéticos, redes neuronales, machine learning, árbol de decisión, red bayesiana, es-sinónimo-de, es-subdisciplina-de, problemáticas, aplicaciones, esta-compuesto-por, es-etapa-de, algoritmos, instrucciones, reglas, entrada, salida, problema, estructura de datos, maldición de la dimensionalidad, sobreajuste, overfitting, minería de datos, bioinformática, diagnóstico médico, reconocimiento óptico de caracteres, reconocimiento de patrones, clasificación de ADN, algoritmo de esperanza y maximización, k means, mapa auto organizado, máquina de vector soporte, support vector machine, aprendizaje por refuerzo, es-estructura-de, es-sinónimo-de, es-entidad-de, es-subárea-de, área, subárea, sinónimo, entidad, aplicaciones, utilidades, dificultades, algoritmos, autores.

**Glosario de términos (Agentes Inteligentes):** estructura de los agentes, agente reactivo simple, agente reactivo basado en modelos, agente basado en objetivos, agente basado en utilidad, agente que aprende, agente de búsqueda en línea, regla de condición acción, reactivo, agente reactivo, estado interno, si-entonces, acción aleatoria, meta, objetivo, submeta, subobjetivo, utilidad, función utilidad, elemento de aprendizaje, elemento de activación, crítica, generador de problema, agente de búsqueda en línea, agente online, sensor, sensar, sensado, simulación, prototipo, entorno cognitivo, cognición, medio ambiente, medio, environment, ambiente, modelado, generador de entorno, actuador, activador, agente, agente inteligente, agente racional, percepción, programa del agente, programación del agente, planner, scheduling, REAS, rendimiento entorno actuadores sensores, agente individual, multiagente, agente de software, softbot, robot, propiedad de entorno, totalmente observable, parcialmente observable, determinista, estocástico, estratégico, cooperativo, competitivo, contínuo, discreto, dinámico, estático, secuecial, episódico, comportamiento, recopilación de información, recuperación de información, omniprescencia, racional, interacción, es-estructura-de, es-sinónimo-de, es-entidad-de, es-subárea-de, área, subárea, sinónimo, entidad, aplicaciones, utilidades, dificultades, algoritmos,autores.

**Glosario de términos (Teoría de Bases de Datos):** teoría de bases de datos, database theory, database, base de dato, tipos de bases de datos, modelo relacional, modelo entidad relación, relación, uno a uno, mucho a muchos, uno a muchos, muchos a uno, clave, clave primaria, superclave, dominio valor, atributo, tabla, tupla, variable, operaciones, unión, selección, proyección, diferencia, producto cartesiano, base de datos de objetos, objeto, encapsulado, orientada a objetos, mensaje, método, herencia, base de datos en red, modelo en red, network model, datos en red, base de datos documental, modelo documental, modelo jerárquico, base de datos jerárquica, sistemas de bases de datos, DB2, Oracle, SQL, MySQL, PostgresSQL, lenguajes de bases de datos, XML, UML, ERWin, arquitectura de bases de datos, arquitectura centralizada, arquitectura cliente servidor, arquitectura paralela, sistemas paralelos, transacción, unidad lógica de trabajo, atomicidad, aislamiento, durabilidad, diseño de base de datos, repetición de información, inconsistencia de datos, primera forma normal, etc. , indexación, indización, es-estructura-de, es-sinónimo-de, es-entidad-de, es-subárea-de, area, subárea, sinónimo, entidad, aplicaciones, utilidades, dificultades, algoritmos,autores.

**Árbol de clasificación de conceptos (taxonomía de conceptos):** se han generalizado los términos que contiene el glosario en los siguiente conceptos, también podemos observar su jerarquía en orden vertical, en la figura 4.1.

**Diccionario de datos:**

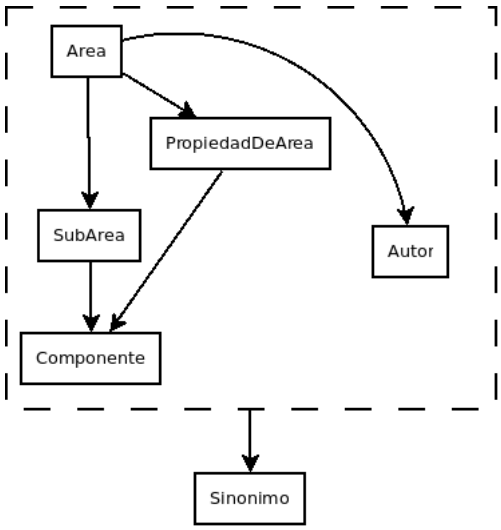



Figura 4.1: Taxonomía de *Conceptos*.

| **Clase** | **Descripción** |
|---|---|
| Área | Subdisciplina del dominio que se está representando con la ontología en general. En este caso el dominio es Ciencias de la Computación y las subdisciplinas (Áreas) son Aprendizaje Automático, Agentes Inteligentes y Teoría de Bases de Datos. |
| PropiedadDeÁrea | Es una propiedad genérica de un Área, esto significa que como propiedad misma es trasladable a otros dominios, como por ejemplo dificultades que se presentan en el Área, algoritmos conocidos, estructuras de datos utilizadas en general en el área, las instancias correspondientes a estas tres situaciones serían *Dificultades*, *Algoritmos* y *Estructuras de Datos*. |
| SubÁrea | Es la subdisciplina dentro del dominio que representa el Área, es menos general que PropiedadDeÁrea porque si tomamos una instancia en particular de SubÁrea no es trasladable a otro dominio, por ejemplo dentro del dominio Aprendizaje Automático, la subárea Aprendizaje por refuerzo. |
| Componente | Parte indivisible de una SubÁrea, cuando decimos indivisible estamos en la base de un nivel de abstracción determinado para definir la ontología porque en caso contrario podríamos explotar en mayor profundidad el componente. |
| Sinónimo | Representa el sinónimo o cómo puede llegar a referirse una frase o término a otra frase o término, se aplica a Área, SubÁrea y Autor. |

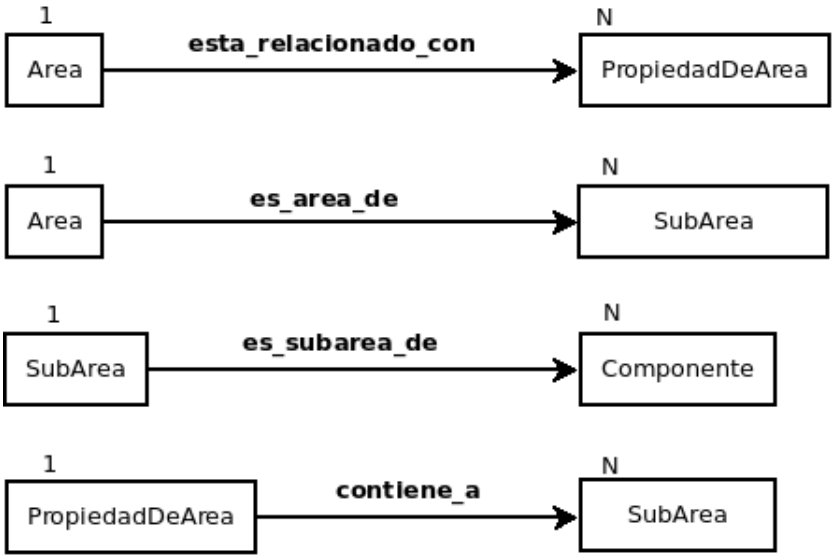
Cuadro 4.2: Diccionario de datos.

**Tabla de instancias:** en esta parte sólo se mostrarán cuatro filas por cada clase, si se desea visualizar el total de las instancias se pueden observar las tablas completas en Apéndice A.

| Concepto | Instancias | Relaciones |
|---|---|---|
| Área | Aprendizaje Automático | es_area_de, contiene_propiedad |
| PropiedadDeÁrea | Dificultades, Aplicaciones, Algoritmos, Componentes. | contiene_subarea, contiene_componente. |
| SubÁrea | Aprendizaje supervisado, Aprendizaje inductivo, Aprendizaje no supervisado, Aprendizaje por refuerzo, Minería de datos. | es_compuesta_por |
| Componente | Overfitting, Maldición de la dimensionalidad, Red neuronal, Proceso gaussiano, Desambiguación. | |
| Sinónimo | Aprendizaje, Learning, ML, Machine Learning. | es_sinonimo_de |

Cuadro 4.3: Tabla de instancias.

**Diagrama de verbos (relaciones):**



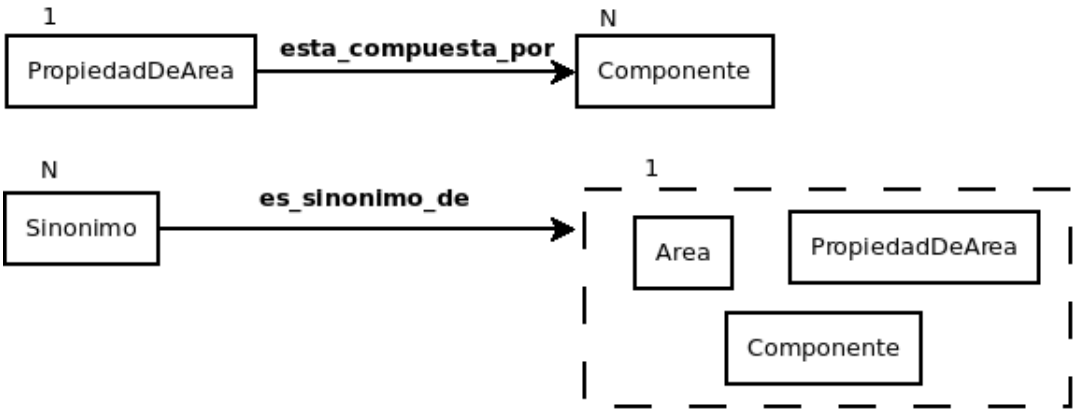



Figura 4.2: Diagrama de verbos.

**Diccionario de verbos:**

| **Verbo (Relación)** | **Descripción** |
|---|---|
| contiene_propiedad | Expresa qué propiedad general contiene el Área. |
| es_area_de | Expresa qué Área contiene a una SubÁrea. |
| es_compuesta_por | Expresa qué *Componente* está dentro de una determinada SubÁrea, la idea es que *Componente* sea un término atómico. |
| contiene_subarea | Expresa qué subáreas son consideradas como tales dentro de una *PropiedadDeArea*. |
| contiene_componente | Igual a es_compuesta_por pero con diferente dominio y rango. |
| es_sinonimo_de | Expresa la relación entre un término y su sinónimo |

Cuadro 4.4: Diccionario de verbos.

### 4.1.4. Integración

Se ha investigado acerca de las ontologías existentes. Actualmente hay partes del dominio dentro de la Inteligencia Artificial que están modelados en una estructura de ontologías pero son insuficientes ya que son muy poco profundas y terminan describiendo sólo generalmente este dominio, además se presenta el problema de que están en idioma Inglés, mientras que aquí se intenta construir una ontología totalmente en español, sólo se incluyen algunos términos en Inglés como instancias de la clase sinónimo.
Con respecto a los términos que se eligen para representar el dominio fueron obtenidos de las fuentes descriptas en la etapa 1), algunos fueron traducidos al español y los términos en los que es poca conocida su traducción se optó por mantenerlos en idioma Inglés.

### 4.1.5. Implementación

La implementación se ha realizado en lenguaje Java, la estructura de datos usada ha sido matrices de enteros que representan las relaciones de la ontología si la observa-

mos como un grafo, también conocida como matriz de incidencia. Donde cada término tiene un número entero asignado y dependiendo del entero que contenga el elemento de la matriz se puede distinguir el tipo de relación que representa de las definidas en la Conceptualización. Además de la matriz de incidencia también se define el arreglo de pesos, donde cada elemento del arreglo es el peso que se le asignó a partir de la función peso seleccionada en la sección siguiente. Dejaremos la parte de pseudocódigo para el capítulo 6, de manera que quede más claro.

Se ha optado por el lenguaje Java porque facilita el proceso previo y posterior que requieren los documentos científicos. En el caso de utilizar otro entorno para implementar la ontología hubiese sido necesario ahondar en APIs existentes para poder articular el formato de exportación de la tecnología utilizada (por ejemplo: Protégé) con Java. En Apéndice B (Diagramas de ontologías), se pueden observar las implementaciones completas de cada área en forma de diagrama.

# Capítulo 5

# Propuesta

Como se ha dicho anteriormente la motivación principal es catalogar documentos científicos en idioma español dentro del área de Inteligencia Artificial, las subáreas de Agentes Inteligentes y Aprendizaje Automático por un lado, y por otro Teoria de Bases de Datos. Se denomina "Vocabularios Controlados" a las ontologías diseñadas en el capítulo 4. Aquí se toma las ontologías como un conjuntos de términos indexados y relacionados sin tener en cuenta el tipo de Concepto (Clase) de cada término.
El objetivo es proveer al usuario final un conjunto de términos jerarquizados según la ontología. Para luego poder construir un repositorio ordenado de documentos o simplemente aplicar un etiquetado sobre ellos.
A continuación se presenta la arquitectura propuesta para llevar a cabo esta tarea (Figura 5.1), observar que Incialización y Armado de vector y asignación de pesos del documento a clasificar (etapa 1) incluyen más de un componente en la Figura 5.1, para indicar esto se ha dibujado los recuadros en líneas de puntos.
Luego se describe en qué consiste cada etapa del proceso de catalogación, para poder tener una idea del flujo por el cual pasará el documento procesado. En cada etapa se presenta primero la parte teórica de la propuesta y luego un breve ejemplo para ir siguiendo en cada una de ellas.
Solamente se han numerados dos etapas para indicar que luego de la Inicialización, se puede iterar sobre la etapa-1 y etapa-2 (en este orden) por cada documento que se desee catalogar. Mientras que la Inicialización solo se realiza por única vez, al menos que se actualice el corpus de documentos para el entrenamiento de la ontología.
Previamente, se presenta un análisis sobre las funciones peso. El rol de la función peso en este proyecto no es primordial, porque el objetivo principal es catalogar documentos científicos. Donde el usuario pueda obtener en primer lugar qué categorías coincidieron con el documento de entrada, y no qué valor númerico de similitud con cierta área (representada en la ontología). Más allá de esta priorización, es necesaria por lo menos una función peso que se comporte bien en este ambiente y justamente es lo que esta sección describe. Vale decir que hasta aquí los términos de los que se disponen son las palabras

claves de los documentos y los términos que aparecen en la ontología.

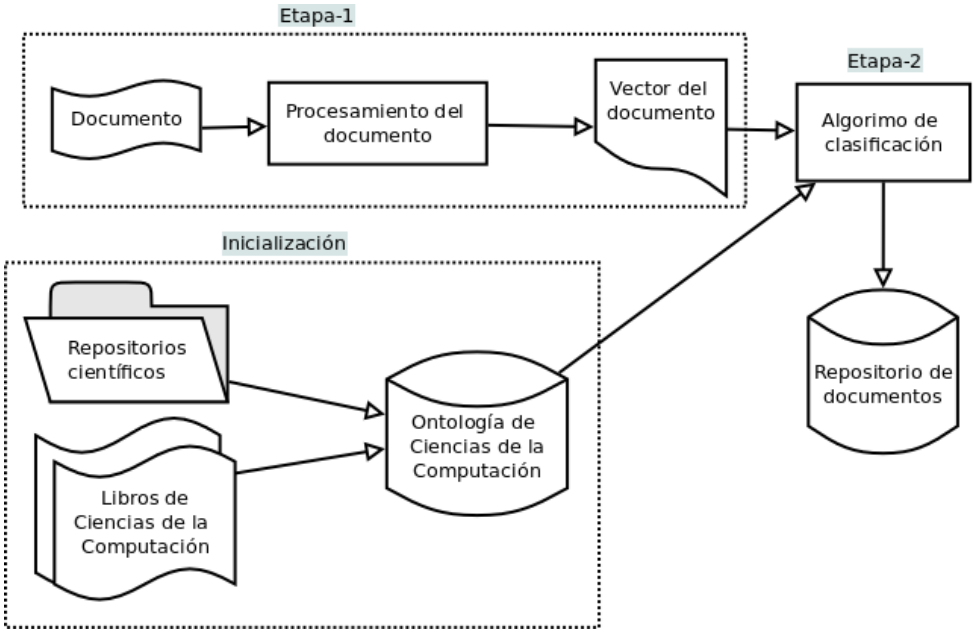



Figura 5.1: Arquitectura propuesta.

## 5.1. Funciones peso

En esta sección se muestra la utilidad de las palabras claves y las ontologías diseñadas, ya que se utilizan como parámetros en la función peso para luego distinguir la importancia de un término con otro término. Es en este punto donde se hace uso del trabajo previo, algunas preguntas que pueden plantearse para enmarcar a este capítulo son: ¿cómo se usarán las palabras claves obtenidas? ¿cómo se utilizarán las ontologías? ¿qué características de la estructura de la ontología son relevantes?, y la más importante ¿por qué es necesaria una función que asigne pesos?.

### Funciones peso para la ontología

En la ontología, más allá de las jerarquizaciones que dan las relaciones entre los términos, es necesario asignar un valor númerico a cada término con el objetivo de distinguir la importancia de uno sobre el otro en el contexto dado por el dominio. Por lo tanto, definimos:

$$P(term_i) = p_i \tag{5.1}$$

donde $i = 1...k, k \in \mathbb{N}$ y $k$ cantidad de términos de la ontología y $p_i$ es el peso que le asignamos al término $i$.
Como posibles parámetros básicos de una posible función peso podríamos considerar

cada término de la ontología, cada relación entre los términos, grado de salida de cada término, frecuencia del término en algún corpus de documentos que sirve como entrenamiento y varios más. En el cuadro 5.1, extraído de [25] y de [9] podemos observar diferentes funciones para asignar pesos a documentos, sin embargo para este caso pueden ser adaptadas para ponderar los términos de la ontología. En [25] y [9] se puede observar el proceso por el cual se llega o se deduce las fórmulas presentadas 5.1.

| Función peso | Parámetro |
|---|---|

$$P_2(t) = \frac{tf \cdot \log \frac{(N}{n)}}{\sqrt{\sum_{OntoTerms}(tf_i \cdot log\frac{N}{n_i})^2}} \quad (5.2)$$

$$P_3(t) = \frac{tf \cdot \log \frac{N}{n}}{\sqrt{\sum_{OntoTerms}(tf_i \cdot log\frac{N}{n_i})^2}} \quad (5.3)$$

$$P_4(t) = 0{,}5 + \frac{0{,}5 \cdot tf}{max(tf)} \quad (5.4)$$

$$P_5(t) = \log \frac{N}{n} \quad (5.5)$$

$$P_6(t) = \frac{tf}{\sqrt{\sum_{OntoTerms}(tf_i)^2}} \quad (5.6)$$

$$P_7(t) = tf \cdot \log \frac{N}{df(t)} \quad (5.7)$$

$$P_8(t) = tf^*(t) \cdot rr(t) \quad (5.8)$$

- $tf$ : Frecuencia del término $t$ en la colección.
- $N$ : Cantidad total de documentos en la colección.
- $n$ : Frecuencia del término $t$ normalizada en la colección.
- $tf^*(t) = 1 + \log tf$
- $rr(t) = 1 + \mu \cdot r_1(t) + \mu^2 \cdot r_2(t)$
- $\mu = \alpha / avlinks$
- $\mu^2 = \alpha^2 / avlinks^2$
- $\alpha$ : factor amortiguación entre las relaciones de longitud 1 y relaciones de longitud 2.
- $avlinks$ : Número promedio de relaciones de longitud 1 en la ontología.
- $avlinks^2$ : Número promedio de relaciones de longitud 2 en la ontología.
- $r_1(t)$ : Relaciones longitud 1 del término $t$ en la ontología.
- $r_2(t)$ : Relaciones longitud 2 del término $t$ en la ontología.
- $df(t)$ : Cantidad de documentos donde el término $t$ aparece al menos una vez.

Cuadro 5.1: Funciones peso

En una mirada general en todas las funciones pueden observarse parámetros comunes como la frecuencia del término en el repositorio, la frecuencia en un documento en particular y la cantidad total de documentos. Mientras que en la funcion 5.8 vemos que se aprovechan las relaciones entre los términos que se dan en la ontología, relaciones de longitud 1 y longitud 2. Además del buen comportamiento de 5.8 descripto en [9],

podemos observar que el marco problemático de la publicación donde surge la función peso es similar al presentado aquí. El marco es el siguiente, se utiliza un tesauro donde se intenta ponderar cada término, se destaca la importancia del uso de un vocabulario controlado en contraste con la asignación de términos libres para la catalogación de un documento. A partir de esos acercamientos con este proyecto es que se selecciona la función y respetando la idea de encontrar una función con buen desempeño en el cálculo de la precisión.
Para esta Tesina se experimentó con la función 5.7 que tiene ciertas coincidencias con las funciones 5.2 a 5.6 y con las función 5.8. De esta forma se probó que con 5.8 se obtuvieron mejores resultados que con la función 5.7, esto se muestra en la Sección 7.
Como definición nos queda la expresión de pesos para cada término de la ontología como:

$$\begin{aligned} P_8(term) &= tf^*(term) \cdot rr(term) \\ &= (1 + \log tf) \cdot (1 + \mu \cdot r_1(term) + \mu^2 \cdot r_2(term)) = p_i \end{aligned} \tag{5.9}$$

donde $term$ es el término en la ontología al que se le va a asignar peso, los otros parámetros están definidos en el Cuadro 5.1.

**Peso del término del documento**

Hasta aquí tenemos la función de pesos para los términos de la ontología. A continuación se necesita calcular el peso de cada documento que se pasa como parámetro para su catalogación. Definimos el vector de términos del documento $Doc$ como:

$$V_{doc} = (term_1, ..., term_k) \tag{5.10}$$

Donde $term_n, n = 1...k, k \in \mathbb{N}$ es una palabra clave del documento $Doc$ obtenida con TexLexAn y que también pertenece como término a la ontología.
En el capítulo siguiente se puede observar que nos quedaremos sólo con los términos de $V_{doc}$ que coinciden con los de la ontología. Sin embargo se nos presenta el problema que quizás el peso de ese término en la ontología no es representativo como peso del documento, es decir, necesitamos una forma de contextualizarlo hacia adentro del documento que se está intentando catalogar. Para lograr esto, lo que se propone es multiplicar la frecuencia del término en el documento con el peso que tiene asignado en la ontología. Por lo tanto, tenemos el peso del documento de la siguiente forma:

$$peso(V_{doc}) = (tf(term_1, Doc) \cdot p_1, ..., tf(term_k, Doc) \cdot p_k) \tag{5.11}$$

donde $tf(term_n, Doc), n = 1...k, k \in \mathbb{N}$ es la frecuencia del término $term_n$ en el documento $Doc$ y $p_n, n = 1...k, k \in \mathbb{N}$ es el peso que tiene asignado el término $term_n$

en la ontología.
Luego se realiza la catalogación del documento a partir de los componentes más pesados del vector $peso(V_{doc})$. Este proceso se puede consultar en los apartados *Asignación de pesos a ontología* (5.2.2) y *Armado de vector y asignación de pesos del documento a clasificar* (5.3).

## 5.2. Inicialización

Se considera el proceso de inicialización la carga de documentos para entrenar la ontología. Y la tarea de asignación de pesos a cada término de la ontología. Se puede observar esta etapa en Figura 5.1. Vale aclarar que el recuadro de la figura con la leyenda 'Libros de Ciencias de la Computación' hace referencia a la utilización de recursos bibliográficos para dar jerarquía a la ontología, como se describió en el Capítulo 4 - Diseño de ontología propuesta.

### 5.2.1. Carga de documentos

Como primer etapa tenemos la carga de documentos en el repositorio, en la Figura 5.1 como 'Repositorios científicos'. Se intenta que los documentos tengan mayoritariamente texto y no demasiadas imágenes ya que en la etapa posterior es necesario que la parte representativa del documento sea su texto y no sus imágenes, ya que estas no se tienen en cuenta para la representación del documento.
El idioma del documento debe ser español y el formato pdf. Tampoco es necesario que posea las palabras claves explícitamente ni el abstract, ya que se evalúa por completo el texto.
Con respecto a la cantidad de páginas, aquí se ha experimentado con documentos que no superen las 50 páginas pero en general no hay restricciones en este aspecto. Se han cargado 120 documentos en total, 40 de cada área.

### 5.2.2. Asignación de pesos a ontología

En primer lugar se genera un vector de términos a partir de la ontología diseñada en el capítulo 4. Se recorre en profundidad el árbol que forma la ontología con los términos como nodos y las relaciones como arcos. Se utiliza el algoritmo clásico para recorrer en profundidad un árbol y se acumulan los términos en el vector que se denomina como $V_{vc}$, donde el subíndice $vc$ es por "Vocabulario controlado". Definimos el vector de términos del vocabulario controlado (VC) como:

$$V_{vc} = (termvc_1, ., termvc_q) \tag{5.12}$$

donde $termvc_n, n = 1...q, q \in \mathbb{N}$, es un término que pertenece a la ontología.
Como disponemos de tres ontologías, tenemos como resultado tres VC:

- $VC_{tbd}$: vector de términos del VC correspondiente al área de Teoría de Bases de datos.
- $VC_{ai}$: vector de términos del VC correspondiente al área de Agentes Inteligentes.
- $VC_{aa}$: vector de términos del VC correspondiente al área de Aprendizaje Automático.

Para ver el contenido completo de cada VC, ver los Apéndices. El paso siguiente es aplicar la función peso del capítulo anterior (ver 5.8) a los VC para asignar un peso a cada término en $VC_{tbd}$ , $VC_{ai}$ , y $VC_{aa}$, tenemos:

$$\begin{aligned} peso(V_{tbd}) &= (P_8(termvcTbd_1), ..., P_8(termvcTbd_m)) \\ &= (tf^*(termvcTbd_1) * rr(termvcTbd_1), ..., tf^*(termvcTbd_m) * rr(termvcTbd_m)) \end{aligned}$$

donde:

$$P_8(term_i) = tf^*(term_i) \cdot rr(term_i)$$

de la misma forma para $VC_{ai}$ y $VC_{aa}$.

$$\begin{aligned} peso(V_{ai}) &= (P_8(termvcAi_1), ..., P_8(termvcAi_n)) \\ &= (tf^*(termvcAi_n) * rr(termvcAi_n), ..., tf^*(termvcAi_n) * rr(termvcAi_n)) \end{aligned}$$

$$\begin{aligned} peso(V_{aa}) &= (P_8(termvcAa_1), ..., P_8(termvcAa_p)) \\ &= (tf^*(termvcAa_1) * rr(termvcAa_1), ..., tf^*(termvcAa_p) * rr(termvcAa_p)) \end{aligned}$$

Donde,

- $termvcTbd_k$, $k = 1..m$, $m \in \mathbb{N}$ es un término en $VC_{tbd}$ y $m$ es la cantidad total de términos en $VC_{tbd}$.
- $termvcAi_k$, $k = 1..n$, $n \in \mathbb{N}$ es un término en $VC_{ai}$ y $n$ es la cantidad total de términos en $VC_{ai}$.
- $termvcAa_k$, $k = 1..p$, $p \in \mathbb{N}$ es un término en $VC_{aa}$ y $p$ es la cantidad total de términos en $VC_{aa}$.

La frecuencia del término que se está calculando es sobre el corpus de documentos de la misma área que sirve como entrenamiento, con un total de 40 documentos como se dijo en la sección 5.2.1. Es la que aparece dentro de dentro de la función $tf^*()$.
Para mostrar un ejemplo, tomaremos el término "transacción" de $VC_tbd$ y se calcula su peso. Este término tiene el índice 140:

$$termvcTbd_{140} = \texttt{`transacción'}$$

aplicando 5.8, se tiene:

$$P_8(termvcTbd_{140}) = tf^*(termvcTbd_{140}) \cdot rr(termvcTbd_{140})$$

donde,

$$\begin{aligned} tf^*(termvcTbd_{140}) &= 1 + log(tf(termvcTbd_{140}, Doc)) \\ &= 1 + log(73) = 1 + 1{,}86 = 2{,}86 \end{aligned}$$

Recordemos que $tf(termvcTbd_{140}, Doc)$ es la frecuencia de $termvcTbd_{140}$ en la colección de documentos (o corpus) sobre Teoría de Bases de datos, en este caso es 73. Para calcular el otro factor de multiplicaccion $rr(termvcTbd_{140})$ debemos calcular $\mu$, $r_1(termvcTbd_{140})$ y $r_2(termvcTbd_{140})$, suponemos $\mu = 0{,}5$ que es un promedio estándar para el factor amortiguación. Y para calcular $r_1(termvcTbd_{140})$ y $r_2(termvcTbd_{140})$ se observa la parte de la ontología donde aparece este término y alguna de sus relaciones:

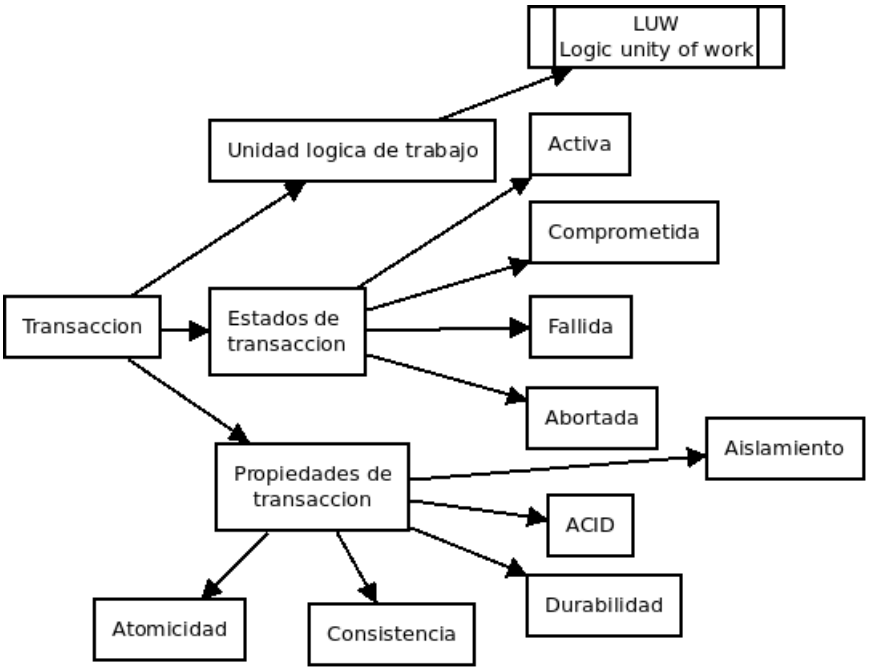



Figura 5.2: Relaciones del término 'transacción'.

Por lo tanto $r_1(termvcTbd_{140}) = 3$ y $r_2(termvcTbd_{140}) = 10$, queda:

$$\begin{aligned} rr(termvcTbd_{140}) &= 1 + (0{,}5 \cdot 3) + (0{,}5^2 \cdot 10) \\ &= 1 + 1{,}5 + 2{,}5 = 5 \end{aligned}$$

El peso final del término 'transacción' es:

$$P_8(termvcTbd_{140}) = tf^*(termvcTbd_{140}) \cdot rr(termvcTbd_{140}) = 2{,}86 \cdot 5 = 14{,}3$$

Este proceso de cálculo de pesos se repite para todos los términos que forman parte de la ontología, al final del proceso tenemos un peso para cada término, para tenerlo en cuenta o no en la siguiente etapa.

## 5.3. Armado de vector y asignación de pesos del documento a clasificar (Etapa 1)

En esta etapa debe seleccionarse el documento que se desea catalogar. Luego se hace del uso del extractor de palabras claves para el procesamiento del documento. Como se ha explicado en el capítulo 2 hemos seleccionado TexLexAn para esta tarea, además de obtener un vector de términos del documento también realiza otras tareas sobre el documento, como resumen, conteo de sílabas, entre otras, aquí sólo utilizaremos los términos retornados para construir el vector de términos del documento. Definimos el vector de términos del documento *Doc* que se obtiene de la salida de TexLexAn como:

$$TexTerms_{doc} = (texterm_1, , texterm_k) \tag{5.13}$$

Donde $texterm_n$, $n = 1...k$, $k \in \mathbb{N}$ es un término del documento *Doc* que se encuentra en la salida de TexLexAn.
Luego se realiza la intersección entre los términos de cada VC ($VC_{tbd}$ , $VC_{ai}$ , y $VC_{aa}$) y los términos de $TexTerms_{doc}$ con el objetivo de filtrar sólo los términos del documento que aparecen en los VC. Tenemos tres vectores resultantes de esta operación:

$$TexTerms_{doc} \bigcap_{stemm} VC_{tbd} = IntTbd_{doc} \tag{5.14}$$

$$TexTerms_{doc} \bigcap_{stemm} VC_{ai} = IntAi_{doc} \tag{5.15}$$

$$TexTerms_{doc} \bigcap_{stemm} VC_{aa} = IntAa_{doc} \tag{5.16}$$

Donde $\bigcap_{stemm}$ compara término a término pero antes de la comparación se lleva cada término a su término raíz, por ejemplo si se compara el término 'operación' con el término 'operaciones', la igualdad de las raíces de los términos es verdadera. Para esto se ha implementado el algoritmo de Porter que permite hacer el stemming de la palabra, es por eso que el símbolo de intersección tiene la palabra stemm como subíndice.
De estos tres vectores sólo se hace uso del vector que más términos coincidieron con uno de los VC, que se lo denomina $Vmax_{doc}$:

$$Vmax_{doc} = Max_{\#}(IntTbd_{doc}, IntAi_{doc}, IntAa_{doc}) \tag{5.17}$$

donde,

$$Max_{\#}(v_1, v_2, v_3) = (\text{vector } v_i / \#v_i > \#v_j \wedge \#v_i > \#v_k) \text{ donde, } i \neq j \neq k \text{ y } j, i, k \in \{1, 2, 3\} \tag{5.18}$$

luego se tiene que,

$$(Vmax_{doc} = IntTbd_{doc}) \vee (Vmax_{doc} = IntAi_{doc}) \vee (Vmax_{doc} = IntAa_{doc}) \quad (5.19)$$

El criterio de selección del vector es que mientras más términos coincidan con el VC, más representativo será el vector obtenido de la intersección, por lo tanto se selecciona el vector de mayor longitud. En el caso que existan longitudes iguales, se opta por el vector de mayor peso.
Recordemos la función peso aplicada a $V_{doc}$ (función 5.11 en el capítulo anterior), sólo que aplicada $Vmax_{doc}$, tenemos:

$$peso(Vmax_{doc}) = (tf(termMax_1, Doc) \cdot ptermMax_1, ..., tf(termMax_k, Doc) \cdot ptermMax_k) \quad (5.20)$$

donde $ptermMax_n$, $n = 1...k$, $k \in \mathbb{N}$, es el peso del término $termMax_n$ en el VC que se ha calculado en la parte de 'Asignación de pesos a ontología' en la inicialización (ver 6.1.2). Y $tf(termMax_n, Doc)$ es la frecuencia de $termMax_n$ en el documento $Doc$. Vale decir que en $Doc$ se cuentan los términos que son raíz del término $termMax_n$ , como se ha utilizado en las intersecciones (5.14), (5.15) y (5.16). Es decir, primero se calcula el stemming (o raíz del término) y luego se cuenta la cantidad de apariciones en el documento $Doc$.
Como ejemplo para esta etapa, se toma como parámetro un documento sobre bases de datos distribuidas. Luego para que no se haga extenso el ejemplo, se observa directamente la intersección de los vectores, nos quedan:

$$\begin{aligned} TexTerms_{doc} \bigcap_{stemm} VC_{tbd} &= IntTbd_{doc} \\ &= \text{(operaciones, transacción, réplica, unión, consulta,} \\ &\quad \text{recuperación, proyección, dbms)} \end{aligned}$$

$$\begin{aligned} TexTerms_{doc} \bigcap_{stemm} VC_{ai} &= IntAi_{doc} \\ &= \text{(comportamiento, continuo, autonomía)} \end{aligned}$$

$$\begin{aligned} TexTerms_{doc} \bigcap_{stemm} VC_{aa} &= IntAa_{doc} \\ &= \text{(aplicaciones, componentes, optimización)} \end{aligned}$$

Se puede observar que las intersecciones coinciden coherentemente con la temática del documento. Se tiene que $Vmax_{doc} = IntTbd_{doc}$ ya que $IntTbd_{doc}$ fue el vector con mayores coincidencias.
Luego se calcula $peso(Vmax_{doc})$, pero antes se listan los valores que toman los parámetros para después reemplazarlos en la aplicación de $peso(Vmax_{doc})$, se tiene que:

| n | $\mathbf{termMax_n}$ | $\mathbf{tf(termMax_n, Doc)}$ | $\mathbf{ptermMax_n}$ |
|---|---|---|---|
| 1 | 'operaciones' | 9 | 35,56 |
| 2 | 'transacción' | 8 | 31.30 |
| 3 | 'réplica' | 10 | 2.38 |
| 4 | 'unión' | 6 | 2.73 |
| 5 | 'consulta' | 4 | 3.58 |
| 6 | 'recuperación' | 3 | 3.26 |
| 7 | 'proyección' | 2 | 2.20 |
| 8 | 'dbms' | 0 | 3.15 |

Cuadro 5.2: Tabla de valores de parámetros.

El peso del vector intersección nos queda:

$$\begin{aligned} peso(Vmax_{doc}) &= (9 * 35{,}56 \ ; \ 8 * 31{,}30 \ ; \ 10 * 2{,}38 \ ; \ 6 * 2{,}73 \ ; \ 4 * 3{,}58 \ ; \\ &\quad 3 * 3{,}26 \ ; \ 2 * 2{,}20 \ ; \ 0 * 3{,}15) \\ &= (320{,}04 \ ; \ 250{,}4 \ ; \ 23{,}8 \ ; \ 16{,}38 \ ; \ 14{,}32 \ ; \ 9{,}78 \ ; \ 4{,}4 \ ; \ 0) \end{aligned}$$

## 5.4. Armado de catalogación del documento (Etapa 2)

Como última etapa nos queda tomar el resultado de la etapa anterior y armar el camino desde cada uno de los primeros seis términos más pesados hacia la raíz de la ontología (aquí volvemos a considerar la ontología como un árbol de términos). Luego unir cada camino para armar una posible catalogación del documento que se ha ingresado como parámetro.
Sean los términos del vector $Vmax_{doc}$ (5.20) de la etapa anterior, se denomina el camino hacia la raíz para cada $termMax_n$ como $getCamino(termMax_n)$. En el cuadro 5.3 se puede observar el algoritmo en pseudocódigo de la función $getCamino()$.
Dentro de los algoritmos aparecen las siguientes funciones a tener en cuenta:

- $preferido(term)$ : retorna el término preferido de $term$. Es decir, la función retorna el término alternativo y/o sinónimo del término $term$.
- $esSinonimo(term$ : retorna *verdadero* si el término $term$ es sinónimo de algún otro término. Si la función retorna *false* se considera $term$ como término preferido.
- $relacion(term_a, term_b)$ : retorna *verdadero* si el término $term_a$ está relacionado jerárquicamente a nivel de la ontología con el término $term_b$.
- $indiceDe(term)$ : retorna el índice que tiene el término $term$ en el VC.

```
Data: term :String, VC : ArrayString
Result: camino : ArrayString
actualIndice, i : Int ;
termPref : String ;
i ← 0 ;
actualIndice ← getIndice(term) ;
/* Se verifica si term es un sinónimo o no                          */
if esSinonimo(term) then
    /* Se agrega el término preferido a camino y actualizamos el
       índice                                                       */
    termPref ← preferido(term) ;
    camino ← camino + termPref ;
    actualIndice ← indiceDe(termPref) ;
else
    /* Si no es sinónimo se agrega el indice actual                 */
    camino ← camino + term ;
end
/* Se recorre el árbol desde el índice actual hasta la raíz         */
while (i <actualIndice - 1 ) ∧ ( i ≠ 0 ) do
    /* Se verifica si VC_i está relacionado con VC_actualIndice     */
    if relacion(i,actualIndice) then
        camino ← camino + VC_i ;
        actualIndice ← i ;
        i ← 0 ;
    else
        i ← i + 1 ;
    end
end
return camino ;
```

Cuadro 5.3: Algortimo que construye el camino a la raíz de un término.

Luego nos queda el algoritmo de armado del árbol de catalogación, denominado ArmarCatalogo(), ver cuadro 5.4.

```
Data: terms, VC : ArrayString
Result: catalogo : Matrix[lenght(VC)][lenght(VC)]
allTerms : ArrayString ;
i, j, len, i1, i2 : Int ;
caminos : Array of ArrayString ;
i ← 0 ;
/* Se calcula el camino a raíz de cada término                    */
while i<length(terms) do
    camino_i ← getCamino(terms_i, VC) i ← i + 1 ;
end
/* Se juntan todos los caminos en allTerms                        */
allTerms ← aplanar(camino) ;
len ← length(allTerms) i ← 0 ;
/* Se compara término a término dentro de allTerms y si están
   relacionados en la ontología se agrega la relación en la matriz
   catalogo                                                       */
while i <len do
    i1 ← indiceDe(allTerms_i) ;
    j ← i ;
    while j <len do
        i2 ← indiceDe(allTerms_j) ;
        if relacion(i1,i2) then
            catalogo_{i1,i2} ← 1 ;
        end
        j ← j + 1 ;
    end
    i ← i + 1 ;
end
return catalogo ;
```

Cuadro 5.4: Algoritmo de catalogación del documento.

Volviendo al ejemplo, siguiendo los pasos de 5.3 y de acuerdo al grafo de Anexo B.4, tenemos:

- $getcamino(termMax_1)$ = ('operaciones', 'modelo relacional', 'tipos de base de datos', 'teoría de bases de datos').
- $getcamino(termMax_2)$ = ('transacción', 'teoría de bases de datos').
- $getcamino(termMax_3)$ = ('réplica', 'sistema distribuido', 'arquitectura de bases de datos', 'teoría de bases de datos').
- $getcamino(termMax_4)$ = ('unión', 'operaciones', 'modelo relacional', 'tipos de base de datos', 'teoría de bases de datos').

- $getcamino(termMax_5)$ = (‘consulta’, ‘lenguajes de bases de datos’, ‘teoría de bases de datos’).
- $getcamino(termMax_6)$ = (‘recuperación’, ‘lenguajes de bases de datos’, ‘teoría de bases de datos’).

Uniendo los caminos a la raíz del ejemplo (siguiendo 5.4) y teniendo en cuenta las relaciones de la ontología nos queda la catalogación siguiente para el documento:

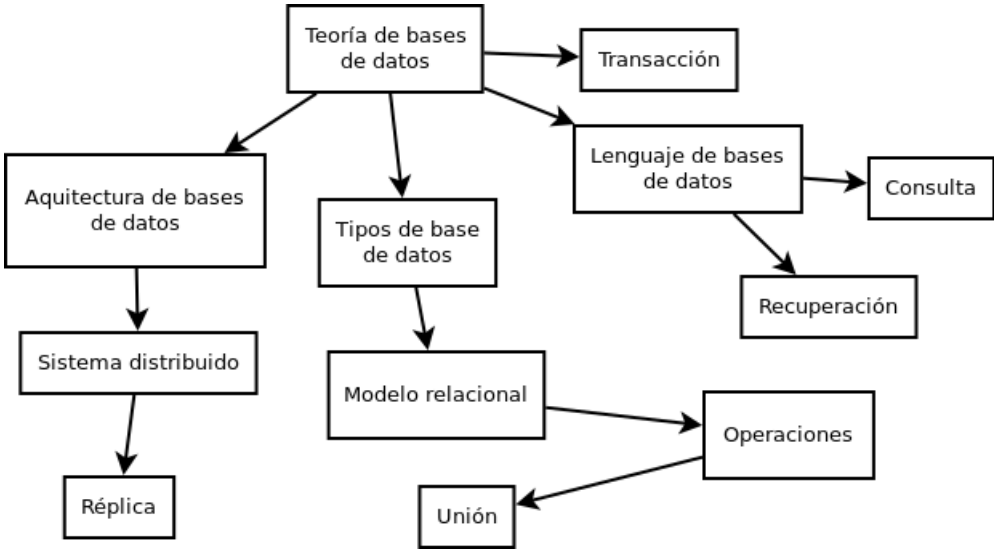



Figura 5.3: Resultado de catalogación del ejemplo

Se han descripto las etapas del proceso de catalogación de un documento, mostrando un ejemplo al final de cada una de ellas. Desde la carga de documentos que luego dan lugar a la asignación de pesos a cada termino de la ontología hasta la etapa que retorna los términos de la catalogación de un documento como resultado. Con estos términos es posible asignar una catalogación jerarquizada sobre cada documento o solamente utilizarlos como simples etiquetas para su posterior búsqueda. También, como se observa en la Figura 5.1, los documentos pueden ser cargados finalmente en un repositorio con el etiquetado de palabras claves usando vocabulario controlado.

# Capítulo 6

# Experimentación y resultados

En esta sección se analizan los resultados obtenidos en la experimentación que tiene como parámetros principales dos corpus que se detallan más adelante. La idea es evaluar la catalogación de los documentos a partir del vocabulario controlado. Se considera una "buena catalogación" si la catalogación automática de un documento se asemeja a la catalogación manual del mismo, es decir a una realizada por una persona que posee conocimientos de las áreas aquí presentadas: Teoría de Bases de Datos, Agentes Inteligentes y Aprendizaje Automático. Para evaluar esto se utiliza el indicador de precisión, su definición en este contexto es:

Precisión: Ratio de documentos catalogados "bien catalogados" de forma automática frente a los documentos que se deberían haber catalogado "bien" de forma manual.

Como guía para lo que sigue se respeta el orden las etapas descriptas y se puede observar que cantidad, tipos de documentos y cómo se ha experimentado la propuesta. A través de gráficos de barra y tablas que muestran los valores concretos que tomaron los parámetros en cada parte de la experiencia.
No se dedicó una sección solamente a los resultados sino que se van evaluando en la medida que se presentan las experimentaciones de cada etapa. Como parte final se describe el trabajo futuro.

## 6.1. Corpus para entrenar vocabulario controlado (Inicialización)

Se han cargado 90 documentos en formato pdf que se dividen en grupos de 30 para cada categoría y han sido utilizados para obtener los pesos de cada término en el vocabulario controlado, es decir para entrenar la misma. La cantidad de palabras de cada documento varía aproximadamente entre 1500 y 7000 , con un promedio de 5200 palabras por cada documento, que serían unas 12 hojas.

## 6.2. Vocabularios controlados y cálculo de peso de los términos (Inicialización)

En esta sección se observan los términos que son tomados desde las ontologías diseñadas en el Capítulo 4, como en ese capítulo no se muestra las ontologías completas se han añadido a la parte de del Apéndice A para consultarlas.
Cada VC tiene un total de 107, 69 y 214 términos respectivamente con $VC_{aa}$, $VC_{ai}$ y $VC_{tbd}$. Se observa una gran diferencia entre $VC_{tbd}$ y los otros dos, lo cual podría suceder que aumente la probabilidad de que un documento a catalogarse sea de Teoría de Bases de Datos. Sin embargo para probar la catalogación, se han seleccionado específicamente documentos de cada área, esto significa que las palabras que contiene cada documento son en su mayoría del área que los expertos considerarían y no de otra.
A continuación se listan los algunos términos de cada vector de los VC:

- $VC_{ai}$ = (agente, entorno, sensor, simulación, prototipo, cognición, cognitivo, medio ambiente, environment, generador de entorno, modelado, actuador, activador, percepción, programa del agente, planeamiento, planner, schedule, scheduling, reas, rendimiento entorno actuadores sensores, agente individual, multiagente, agente software, softbot, robot, agente de software, propiedades de entorno, episódico, secuencial, estático, dinámico, discreto, continuo, competitivo, cooperativo, colaborativo, estratégico, estocástico, determinista, observable, ...)

- $VC_{aa}$ = (aprendizaje automático, machine learning, dificultades, overfitting, sobreajuste, maldición de la dimensionalidad, efecto hughes, curse of dimensionality, dimensionalidad, aplicaciones, conexionismo, bioinformática, biomedicina, red neuronal, minería de datos, data mining, árbol de desición, diagnóstico médico, reconocimiento óptico de carácteres, ocr, optical character recognition, reconocimiento de patrones, patrón, clasificación de adn, clasificación de secuencias de adn, adn, secuencia de adn, motor de búsqueda, reconocimiento del habla, robótica, generalización, ...)

- $VC_{tbd}$ = (teoría de bases de datos, database, bases de datos, tipos de base de datos, modelo de base de datos, modelo jerárquico, modelo jerárquico de base de datos, base de datos jerárquica, hierarchical database model, base de datos documental, documental database, modelo documental, modelo en red, network model, datos en red, base de datos de objetos, objeto, herencia, método, mensaje, encapsulado, clase, lenguaje de programación persistente, object database, orientada a objetos, ...)

Con respecto a la asignación de pesos de cada término, se ha experimentado con las dos funciones de peso descriptas en las secciones anteriores, recordamos (5.7) y (5.8):

$$P_7(t) = tf \cdot \log \frac{N}{df(t)} \tag{5.7}$$

$$P_8(t) = tf^*(t) \cdot rr(t) \tag{5.8}$$

Algunos de los pesos de cada término en cada VC quedan distribuidos como se muestra en los Cuadros (6.2), (6.1) y (6.3).

| **Agentes Inteligentes** | | | | |
|---|---|---|---|---|
| **Índice** | **Término** $t$ | $P_7(t)$ | **Término** $t$ | $P_8(t)$ |
| 1 | agente | 3116 | agente | 101,89 |
| 2 | meta | 541 | entorno | 92,12 |
| 3 | modelado | 398 | estructura de los agentes | 20,29 |
| 4 | multiagente | 306 | propiedades de entorno | 12,18 |
| 5 | activador | 248 | agente que aprende | 5,59 |
| 6 | entorno | 206 | agente basado en objetivos | 4,58 |
| 7 | comportamiento | 196 | meta | 4,05 |
| 8 | aprendizaje | 180 | agente software | 3,94 |
| ⋮ | | | | |

Cuadro 6.1: Pesos de términos en $VC_{ai}$.

| **Aprendizaje Automático** | | | | |
|---|---|---|---|---|
| **Índice** | **Término** $t$ | $P_7(t)$ | **Término** $t$ | $P_8(t)$ |
| 1 | aprendizaje | 773 | aprendizaje automático | 129,57 |
| 2 | minería | 620 | aplicaciones | 113,49 |
| 3 | robótica | 537 | componentes | 57,17 |
| 4 | algoritmo | 391 | aprendizaje supervisado | 35,90 |
| 5 | clasificador | 385 | algoritmo | 29,57 |
| 6 | clasificación | 385 | clustering | 26,18 |
| 7 | refuerzo | 368 | aprendizaje por refuerzo | 23,96 |
| 8 | aplicaciones | 304 | aprendizaje no supervisado | 16,32 |
| ⋮ | | | | |

Cuadro 6.2: Pesos de términos en $VC_{aa}$.

| Teoría de Bases de Datos | | | | |
|---|---|---|---|---|
| **Índice** | **Término** $t$ | $P_7(t)$ | **Término** $t$ | $P_8(t)$ |
| 1 | relación | 546 | modelo relacional | 76,55 |
| 2 | objeto | 503 | teoría de bases de datos | 47,62 |
| 3 | bases de datos | 347 | transacción | 31,30 |
| 4 | atributo | 318 | tipos de base de datos | 29,41 |
| 5 | clase | 269 | indexación | 28,58 |
| 6 | clave | 232 | índice | 28,56 |
| 7 | consulta | 232 | sistemas de bases de datos | 25,17 |
| 8 | entidad | 229 | normalización | 21,02 |
| ⋮ | | | | |

Cuadro 6.3: Pesos de términos en $VC_{tbd}$.

Analizando los Cuadros (6.1), (6.2) y (6.3) se observa que la indización mejora con $P_8(t)$ ya que las frases más representativas de cada VC toman un mayor peso mientras que los términos individuales bajan en la tabla, en comparación con $P_7(t)$. Por ejemplo la frase "aprendizaje automático" sube del índice 11 al 1 mientras que el término "aprendizaje" baja del índice 1 al 17.
Como las frases o términos con más peso son los mas representativos con $P_8(t)$ luego tienen mayor probabilidad de ser tomados para la catalogación, por lo tanto se utiliza $P_8(t)$.
Además en las dos funciones aparece el cálculo de frecuencia del término al que se le quiere asignar peso, en este punto se ha experimentado de dos formas. La primera consiste en comparar por igualdad de strings (cadena de carácter) el término del VC con cada término en el documento de entrenamento mientras que la segunda forma es comparar el stem (o raíz) de los dos términos. Evidentemente la segunda forma es mejor y es por la que se ha optado. Ya que por ejemplo en la comparación por igualdad la frase 'bases de datos' es distinta a 'base de datos'.

## 6.3. Vectores y asignación de peso a los documentos (Etapa 1 y 2)

Aquí aparece el segundo corpus que está compuesto por 90 documentos divididos en grupos de 30 para cada área.

- Documentos AI: desde documento 1 a documento 30.
- Documentos AA: desde documento 31 a documento 60.
- Documentos TBD: desde documento 61 a documento 90.

Han sido utilizados en el prototipo para testear la catalogación. A diferencia de la utilización del corpus anterior, en este sí se extraen términos con TexLexAn para luego

armar el vector de términos de cada documento $TexTerms_{doc}$ (ver 5.13). La cantidad de términos del vector de términos que retorna la ejecucción del extractor TexLexAn son las siguientes, en 5.13 se encuentra nombrado como $k$ :

| **Área** | $k$ **: longitud de** $TexTerms_{doc}$ |
|---|---|
| Agentes Inteligentes (AI) | $45 \leq k \leq 460$, gráfico 6.1 |
| Aprendizaje Automático (AA) | $32 \leq k \leq 370$, gráfico 6.2 |
| Teoría de Bases de Datos (TBD) | $50 \leq k \leq 160$, gráfico 6.3 |

Cuadro 6.4: Cotas de longitud de $TexTerms_{doc}$.

A continuación se puede observar como varía $k$ de $TexTerms_{doc}$ para cada documento. Expresado en gráficos de barras por áreas, (6.2),(6.1) y (6.3). Lo deseado es que en general se mantenga un $k$ bastante uniforme, para que los vectores $TexTerms_{doc}$ tengan la misma cantidad de términos, esta idea es más fuerte en las áreas de AA y TBD.

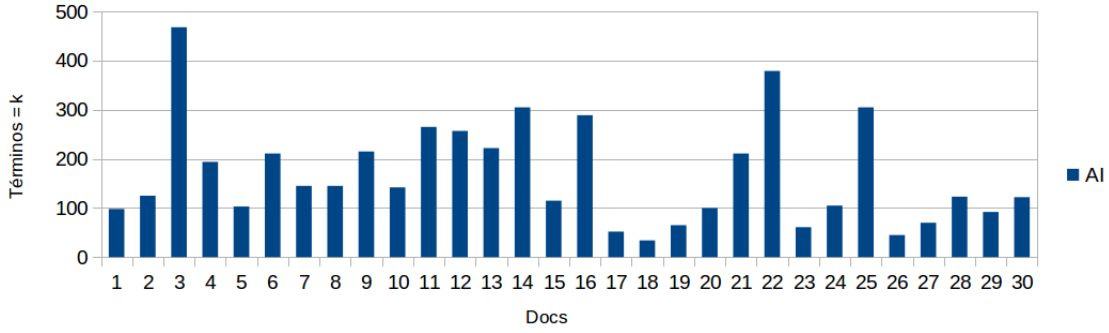



Figura 6.1: Longitud de vectores de documentos en AI

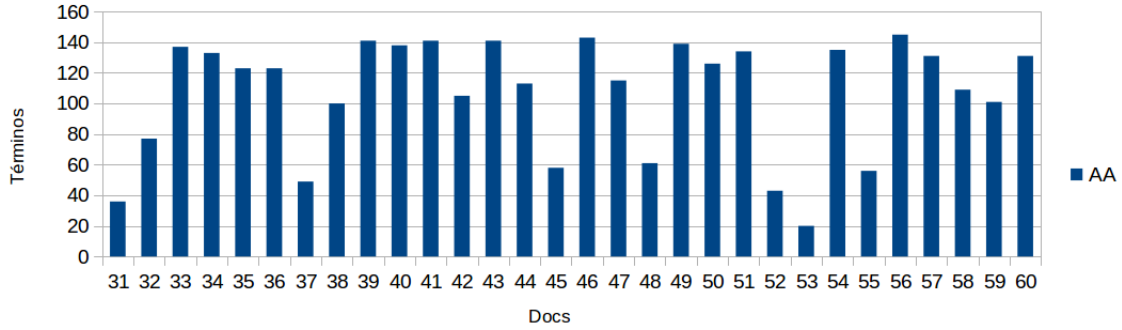



Figura 6.2: Longitud de vectores de documentos en AA

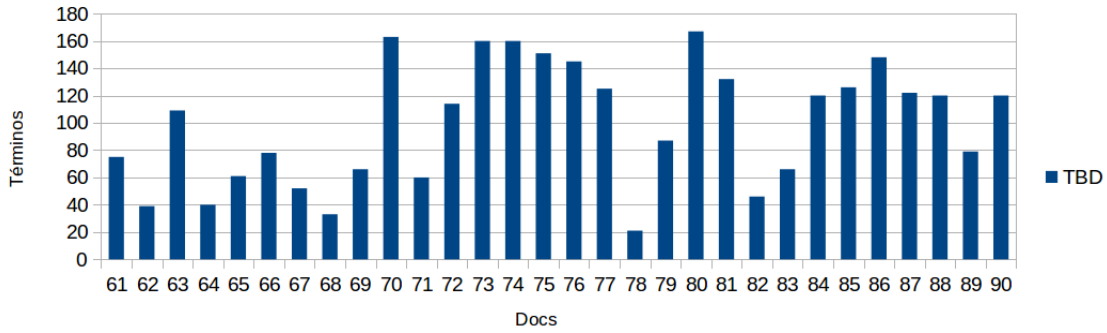



Figura 6.3: Longitud de vectores de documentos en TBD

Luego se hace la intersección de estas palabras claves con las palabras que componen los VC de cada área lo que disminuye considerablemente el conjunto de palabras, como se vió en las definiciones (5.16), (5.15) y (5.14).
Como se sabe al llevar a cabo una intersección es necesario una forma de comparar término a término para comparar la igualdad de los mismos. Para esto se ha experimentado con tres tipos de intersecciones:

1. La primera consiste en comparar los términos por igualdad exacta del término, se simboliza $\bigcap$.
2. La segunda es aplicando stem a cada término que se compara, se simboliza $\bigcap_{stemm}$, ya nombrada en el capítulo anterior.
3. Implementando la distancia de Levenshtein igual a 2 entre los términos, sin embargo esta se ha descartado inmediatamente después de la implementación porque los términos que se consideraban "cercanos" no lo eran y como consecuencia no eran representativos del documento que se estaba representando.

A continuación se muestra en gráfico de barras el resultado de las intersecciones (usando $\bigcap_{stemm}$) con cada VC, luego se analizan las intersecciones usando 1. y 2. y se observa que 2. es conveniente.

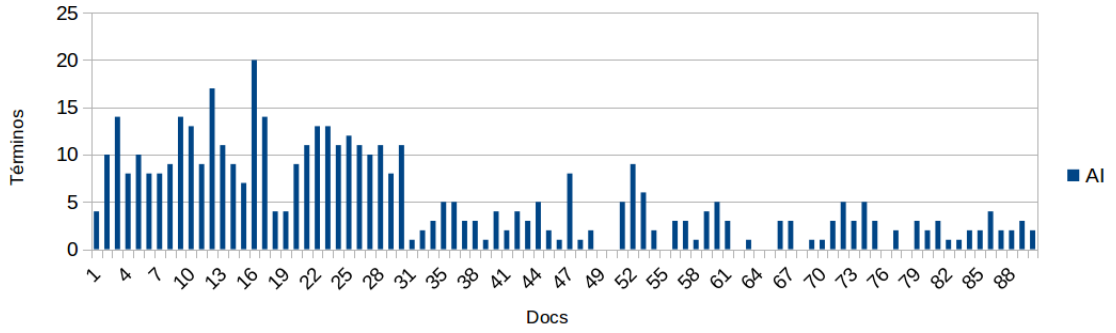



Figura 6.4: Longitud de vectores de intersecciones $IntAi_{doc}$.

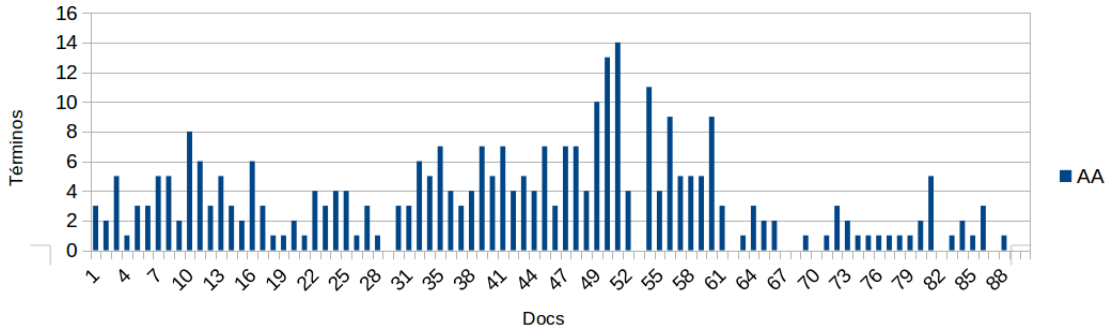



Figura 6.5: Longitud de vectores de intersecciones $IntAa_{doc}$.

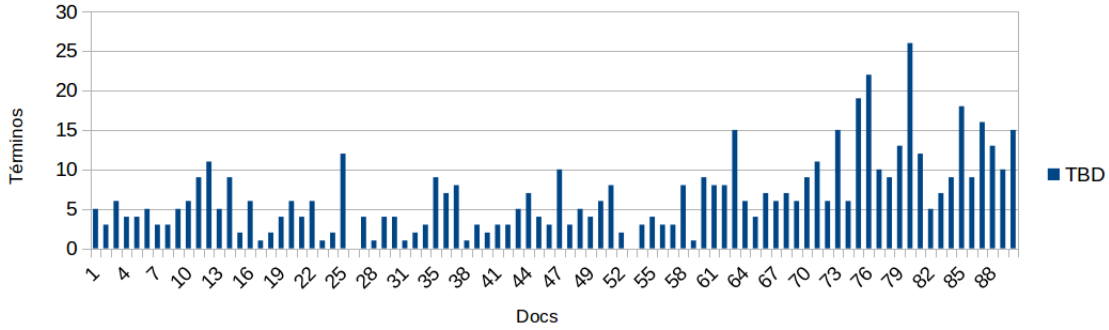



Figura 6.6: Longitud de vectores de intersecciones $IntTbd_{doc}$.

En la figura (6.4) se observa que del documento 1 al documento 30 la cantidad de términos al realizar la intersección es mayor a los documentos desde el 31 al 60. Este es el resultado deseado ya que del 1 al 30 son del área AI y se ha realizado la intersección con $VC_{ai}$. De la misma forma que la figura (6.4) sucede con las figuras (6.5) y (6.6). Mas abajo, en las figuras (6.7), (6.8) y (6.9) se muestra la comparación entre las intersecciones $\bigcap$ y $\bigcap_{stemm}$ de cada vector intersección de cada documento. Vale decir que es una comparación a nivel cantidad de términos iguales, es de interés que coincidan la mayor cantidad de términos posibles siempre que se mantengan dentro del VC.

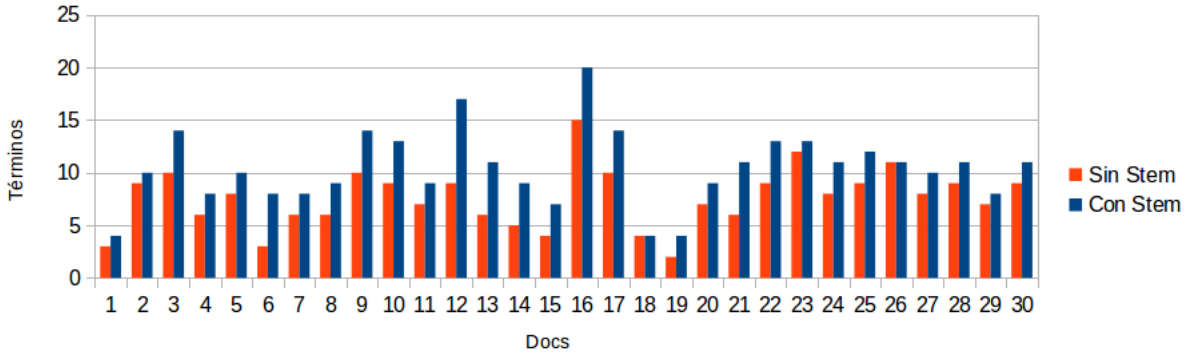



Figura 6.7: Comparación de vectores con $\bigcap$ y $\bigcap_{stemm}$ en AI.

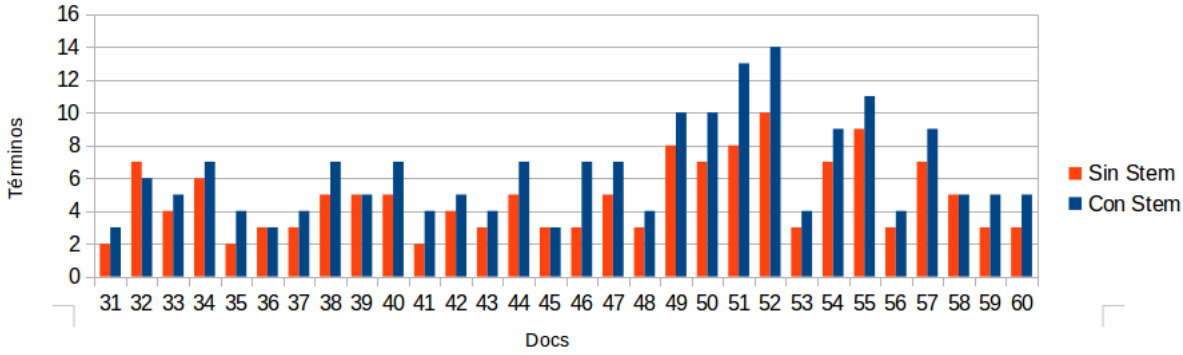



Figura 6.8: Comparación de vectores con $\bigcap$ y $\bigcap_{stemm}$ en AA.

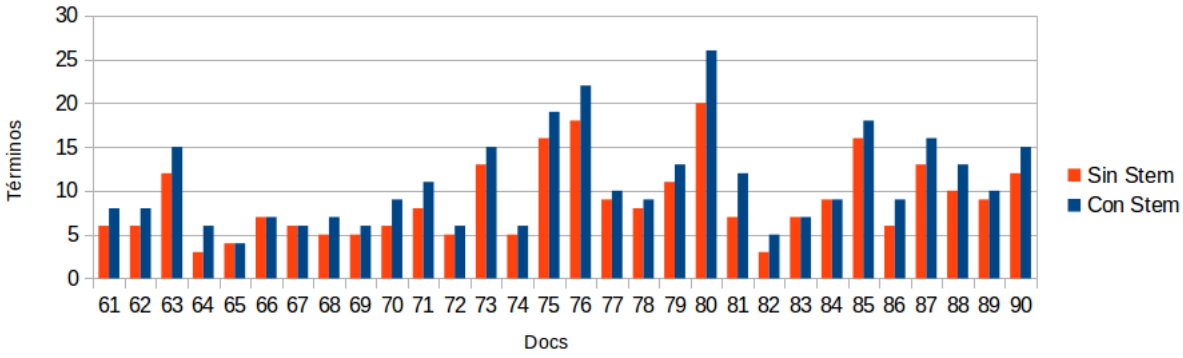



Figura 6.9: Comparación de vectores con $\bigcap$ y $\bigcap_{stemm}$ en TBD.

Se puede observar que en estos tres gráficos con $\bigcap_{stemm}$ se obtienen vectores de mayor longitud, esto significa mayor precisión ya que los términos que coinciden son más, ver la justificación luego de figura (6.5) donde se explica esto.
El cálculo de precisión es:

$$precision = \frac{\text{cant. términos coincidente con asignación manual}}{\text{cant. términos de asignación manual}} \tag{6.3}$$

Se muestran algunos buenos ejemplos de cada VC a continuación:

| **VC** | **Doc** | **Asig. manual** | **Int.$\bigcap$** | **Precisión $\bigcap$** | **Int.$\bigcap_{stemm}$** | **Precisión $\bigcap_{stemm}$** |
|---|---|---|---|---|---|---|
| AI | 8 | aprendizaje, medio ambiente, multiagente, meta, cooperativo, interacción | aprendizaje, medio ambiente, interacción, multiagente, adaptación, dinámico | $\frac{4}{6} = 0,666$ | adaptación, activador, multiagente, aprendizaje, interacción, cooperativo, medio ambiente, dinámico | $\frac{5}{6} = 0,833$ |
| AA | 19 | aprendizaje automático, aprendizaje por refuerzo, proceso de decisión de markov, algoritmo, proceso estocástico | procesos de decisión de markov, aprendizaje automático, algoritmo, maximizar, proceso estocástico | $\frac{4}{5} = 0,8$ | algoritmo, óptimo, aprendizaje por refuerzo, aprendizaje automático, maximizar, proceso estocástico, reconocimiento de patrones, procesos de decisión de markov] | $\frac{5}{5} = 1$ |
| TBD | 20 | integridad y seguridad, integridad referencial, disparador, autorización, restricción de dominio, asertos, microsoft sql server, rol, privilegios, relación, tabla, entidad | autorización, microsfot sql server, integridad y seguridad, restricción de dominio, disparador, relación, tupla, atributo, privilegio, borrar, dominio, clave, valor, inserción, modificación, tabla, entidad, consulta, transacción | $\frac{7}{12} = 0,583$ | relación, clave, autorización, operaciones, modificación, atributo, disparador, restricción de dominio, transacción, tupla, actualizar, valor, integridad y seguridad, borrar, privilegio, tabla, dominio, insertar, consulta, entidad, clave primaria, asertos, sql, indización | $\frac{10}{12} = 0,833$ |

Cuadro 6.5: Cálculo de precisióncon $\bigcap$ y $\bigcap_{stemm}$.

En la Tabla 6.5 se tiene la primer columna que hace referencia al VC en el que se encuentra el documento a clasificar, luego el índice del documento en el corpus. En la tercer columna la asignación manual de términos del documento, es decir asignados

por el lector del documento. En las columnas restantes estan los términos asignados automáticamente, utilizando $\bigcap$ y $\bigcap_{stemm}$ y sus respectivas precisiones. Es importante aclarar que estos términos no son la catalogación final del documento.
Además en la Tabla 6.5 se puede observar cómo aumenta la precisión al utilizar $\bigcap_{stemm}$. En general para todos los documentos a clasificar que forman parte del corpus aumentan su precisión, esto se debe a que el vector obtenido es de mayor longitud, aplicando igualdad de términos con stemming. Al ser de mayor longitud aumentan las concidencias con la asignación manual de términos.
Luego de calcular la precisión para todos los documentos se obtuvieron los siguientes promedios como resultado:

| **VC** | **Promedio de precisión** $\bigcap$ | **Promedio de precisión** $\bigcap_{stemm}$ |
|---|---|---|
| AI | 0,692 | 0,753 |
| AA | 0,603 | 0,649 |
| TBD | 0,562 | 0,622 |
| **TOTAL** | 0,619 | 0,674 |

Cuadro 6.6: Promedios de precisión.

En esta última tabla puede observarse una pequeña mejora de la precisión utilizando $\bigcap_{stemm}$. Luego se llevaría a cabo la etapa de catalogación a partir de cada vector intersección. Sin embargo ya se ha utilizado precisión para evaluar estos últimos resultados, a nivel asignación de términos y no de la catalogación. Se recuerda que el proceso de catalogación se describe en la Sección 6.3. Donde se presenta el algoritmo para realizar la misma y un breve ejemplo.
Al evaluar la precisión obtenida se considera un buen y deseable desempeño de la implementación de la propuesta. Por lo que es esperable que la precisión se incremente. Porque al realizar la catalogación de los restantes documentos a partir del vector que genera $\bigcap_{stemm}$ se agregan términos a los que actualmente se tienen y esto implica mayor probabilidad de coincidencias.

# Capítulo 7

# Conclusiones

Se han desarrollado Ontologías para su utilización como vocabulario controlado y un sistema catalogador de documentos que utiliza el mismo VC. Al principio se presentaron algunos extractores de palabras claves, realizando una comparación con las características de cada uno y optando por TexLexAn, la mejor opción para este caso.
Luego se repasaron las diferentes definiciones de lo que se considera 'Ontología' dentro de las Ciencias de la Computación, y se describieron en detalle algunas de las metodologías de diseño existentes para ontologías. Se compararon las metodologías teniendo en cuenta sus ventajas y desventajas de cada una, optando por Methontology. Se ha aplicado paso a paso esta metodología para definir tres áreas del dominio: Agentes Inteligentes, Aprendizaje Automático y Teoría de Bases de Datos.
Brevemente se listaron algunas funciones para ponderar los términos en la ontología, seleccionado la función que mejor aprovecha la estructura de la ontología.
En la parte de la propuesta se detallaron las etapas en un marco teórico, es decir, con sus respectivas fórmulas, parámetros y algoritmos para casos generales y no solo para el prototipo experimentado.
Con respecto a la experimentación, se ha implementado un prototipo de la propuesta en lenguaje Java. El balance del comportamiento es positivo ya que se obtuvieron las catalogaciones deseadas. Se ha calculado la precisión para cada área con buenos resultados, con una precisión promedio igual a 0.674 cuando se usa stemming, superando [9] y [19]. Brevemente, en [9] se tiene una conjunto de 258 documentales de TV donde cada uno tiene asociado 362 documentos (en la publicación se los llama documentos de contexto) y por otro lado se tiene un tesauro llamado GTAA (la sigla en español es, *Tesauro en Común de Archivos Audiovisuales*). El caso de estudio es asignar palabras claves a cada programa de TV utilizando el tesauro como VC, al comparar la asignación de palabras claves de forma manual con la asignación de palabras claves automática, se obtienen precisiones entre 0.35 y 0.44, dependiendo de que función peso se utilice para los términos. Mientras que en [19], los documentos se obtienen de forma aleatoria de FAO (la sigla en español, *Organización de Comida y Agricultura*) perteneciente a

Naciones Unidas y se utiliza el tesauro llamado Agrovoc. Se asignan términos claves a cada documento utilizando como VC el tesauro y KEA [7]. Se obtiene una precision de 0.0134 con respecto a la asignación de PC utilizando como parámetro el texto completo del documento y 0.88 con respecto a la asignacion de PC utilizando como parámetro el abstract del documento. Se recuerda que en esta propuesta de Tesina, la asignacion de terminos claves se realiza sobre todo el texto que compone al documento.

Como trabajo futuro, son variadas las partes de los posibles disparadores y puertas abiertas que deja la propuesta.

Siguiendo el orden en que se fueron presentando las diferentes tecnologías y estructuras de datos. En el extractor de términos existe un desventaja. TexLexAn arroja como resultados un vector donde cada componente es un término y no una frase. Esto lleva a enriquecer la ontología con términos (un sólo string) que hagan referencia a frases para que luego en la catalogación se obtenga la frase a la que refiere el término. Se podría seleccionar un mejor extractor de términos donde esto no ocurra.

Con respecto a la parte de ontologías se podría implementar la misma en alguna plataforma que actualmente funcione bien, como Protégé, OntoEdit u otra herramienta. Con esta implementación se podría hacer la ontología más flexible y reutilizable a la hora de enriquecerla. Recordemos que en este proyecto se ha implementado directamente en Java con estructura de arreglos y matrices. Si se implementara en algunas de las plataformas nombradas se abre un desafío de poder articular entre la plataforma de la ontología y Java, ya que en este último están implementados todos los algoritmos.

También hacia el interior de la ontología se puede implementar un Concepto o Clase nueva que sea *Autor*. Dentro de cada área existen autores que son referentes de las mismas como puede ser Peter Norvig y Stuart J. Russell para la Inteligencia Artificial.

Con respecto a las funciones peso se podría experimentar con las demás funciones, ya que aquí solo se han implemetado dos de ellas y evaluar resultados. Podría buscarse una función que contextualice mejor un término con respecto a un VC, ya que para hacer esto se ha recurrido a la frecuencia del término en el documento a clasificar.

En cuanto al proyecto general sería de gran utilidad implementarse en un repositorio de documentos, por ejemplo educativos o de tesinas. Donde el usuario que sube los documentos al servidor que contiene el repositorio no tenga el conocimiento y los saberes necesarios para la catalogación del documento en el momento de la carga. De modo que esta se realizaría de manera automática.

Hay que tener en cuenta que para implementar esta última idea será necesario diseñar los ontologías correspondientes a los dominios que darían un marco a los documentos a catalogar.

# Apéndice A

# Tablas de instancias

## A.1. Instancias de Agentes Inteligentes, ver cuadro A.1

| **Agentes Inteligentes** | | |
|---|---|---|
| **Concepto** | **Instancias** | **Relaciones** |
| Área | Agentes | es_area_de, esta_relacionado_con |
| PropiedadDeÁrea | Entorno, propiedad del entorno, comportamiento, estructura de los agentes | esta_relacionado_con, contiene_a |
| SubArea | Agentes individuales, multiagente, agente software, agente de búsqueda en línea, agente que aprende, agente basado en utilidad, agente basado en objetivos, agente reactivo, agente racional. | es_area_de, es_subarea_de |
| Componente | Sensor, simulación, prototipo, cognición, medio ambiente, generador de entorno, actuador, percepción, programa del agente, REAS, softbot, observable, determinista, estocástico, estratégico, cooperativo, competitivo, contínuo, discreto, dinámico, estático, secuencial, episódico, recuperación de información, omnipresencia, autonomía, racionalidad, adaptación, interacción, exploración, generador de problema, problema, crítica, aprendizaje, elemento de aprendizaje, utilidad, meta, submeta. | es_subarea_de, es_sinonimo_de |
| Sinónimo | Cognitivo, modelado, planeamiento, planner, schedule, scheduling, rendimiento entorno actuadores sensores, agente de software, robot, colaborativo, recuperación de la información, interactuar, online, objetivo, subobjetivo. | es_sinonimo_de |

Cuadro A.1: Tabla de instancias de Agentes Inteligentes.

## A.2. Instancias de Aprendizaje Automático, ver cuadro A.2

## A.3. Instancias de Teoría de Bases de Datos, ver cuadro A.3

| Aprendizaje Automático | | |
|---|---|---|
| **Concepto** | **Instancias** | **Relaciones** |
| Área | Aprendizaje Automático | es_area_de, esta_relacionado_con |
| PropiedadDeÁrea | Aplicaciones, algoritmo, dificultades, componentes. | esta_relacionado_con, contiene_a |
| SubArea | Minería de datos, bioinformática, reconocimiento de patrones, motor de búsqueda, robótica, clasificación, recuperación de información, optimización, clustering, aprendizaje supervisado, aprendizaje inductivo, aprendizaje no supervisado, aprendizaje por refuerzo, aprendizaje semi-supervisado. | es_area_de, es_subarea_de |
| Componente | Arbol de decisión, conexionismo, biomedicina, diagnóstico médico, reconocimiento óptico de caracteres, patrón, reconocimiento del habla, generalización, predicción, proceso gaussiano, desambiguación, similitud, multiplicador de Lagrange, vecinos más cercanos, algoritmo genético, algoritmo esperanza maximización, minimizar, maximizar, mapa autoorganizado, k means, maldición de la dimensionalidad, overfitting, árbol de decisión, maquina de vector soporte, red neuronal, backpropagation, perceptron, transducción, compensación por explotación, compensación por exploración, procesos de decisión de Markov, proceso estocástico. | es_subarea_de, es_sinonimo_de |
| Sinónimo | Machine learning, aprendizaje, learning, diagnóstico, OCR, optical character recognition, clasificación de secuencias de ADN, ADN, secuencias de ADN, reconocimiento, Gaussian process, categorizador, clasificador, information retrieval, semejanza, Lagrange, óptimo, algoritmo vecinos más cercanos, k vecinos más cercanos, k NN, algoritmo de clustering, minimización, maximización, expectation maximization, expectativa de maximización, esperanza maximización, self autoorganizing maps, efecto hughes, curse of dimensionality, dimensionalidad, sobreajuste, support vector machine, SVM, máquina vector soporte, retro propagación, backward propagation, programación lógica inductiva, aprendizaje reforzado, reinforcement learning, exploration trade off, exploitation trade off, estocástico, Markov decision process, MDP. | es_sinonimo_de |

Cuadro A.2: Tabla de instancias de Aprendizaje Automático.

| Teoría de Bases de Datos | | |
|---|---|---|
| **Concepto** | **Instancias** | **Relaciones** |
| Área | Teoría de bases de datos | es_area_de, esta_relacionado_con |
| PropiedadDeÁrea | Dificultades, arquitectura de bases de datos, tipos de bases de datos, propiedades de transacción, estados de transacción | esta_relacionado_con, contiene_a |
| SubArea | Diseño de bases de datos, dependencia funcional, integridad y seguridad, arquitectura cliente servidor, sistema paralelo, sistema distribuído, lenguaje de bases de datos, indexación, sistema de bases de datos, modelo entidad relación, modelo relacional, modelo en red, bases de datos documental, bases de datos de objetos, modelo jerárquico. | es_area_de, es_subarea_de |
| Componente | relación, uno a uno, uno a muchos, muchos a muchos, muchos a uno, clave primaria, super clave, clave candidata, atributo, atributo multivalorado, dominio, valor, entidad, tabla, tupla, variable, ejemplar de relación, operaciones, unión, selección, proyección, composición, diferencia, producto cartesiano, modificación, borrar, actualizar, insertar, objeto, mensaje, método, herencia, herencia múltiple, lenguaje de programación persistente, postgressql, mysql, microsoft sql server, sybase, informix, datalog, query by example, oracle, db2, erwin, xml, etiqueta, marca, uml, visio enterprise, relational rose, consulta, recuperación, arquitectura centralizada, técnicas de división, turno rotatorio, división por asociación, división por rangos, grano grueso, grano fino, homogéneas, heterogéneas, réplica, transacción global, transacción local, protocolo de consenso de quorum, protocolo sesgado, protocolo de bloqueo, protocolo de mayoría, integridad referencial, asertos, disparador, restricción de dominio, evento condición acción, autorización, privilegio, rol, unidad lógica de trabajo, atomicidad, copia en la sombra, consistencia, aislamiento, durabilidad, activa, abortada, fallida, parcialmente comprometida, normalización, primera forma normal, segunda forma normal, forma normal Boyce Codd, tercera forma normal, cuarta forma normal, forma normal de reunión por proyección, forma normal de dominio clave, regla de reflexividad, regla de transitividad, regla de aumentatividad, regla de descomposición, recubrimiento canónico, imposibilidad de información, repetición de información, índices, índices ordenados, | |

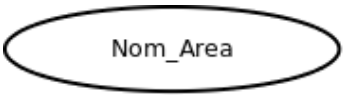
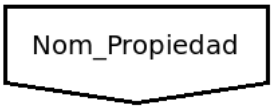
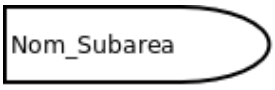
| | | |
|---|---|---|
| | asociación dinámica, asociación estática, función hash, arbol b+, índices asociados, índice secundario, índice multinivel, índice disperso, índice denso, índice primario, clave de búsqueda | es_subarea_de, es_sinonimo_de |
| Sinónimo | base de datos, database, database theory, modelo de base de datos, entity relationship model, base de datos entidad relación, entidad relación, uno a varios, varios a varios, varios a uno, superclave, base de dato relacional, relational model, databases relational, inserción, object database, base de datos orientada a objetos, orientada a objetos, encapsulado, clase, programación persistente, base de datos en red, network model, database network model, datos en red, datos de red, documental database, modelo documental, modelo jerárquico de base de datos, base de datos jerárquica, hierarchical database model, dbms, sgbd, qmf, qbe, sql, acces, lenguaje, query, visio, markup, extensible markup language, database architecture, system architecture, system architecture of database, cliente servidor, arquitectura paralela, distribuída homogénea, distribuída homogénea, duplicado, restricción, trigger, integridad de datos, seguridad de datos, integridad, seguridad, grant, autorización de base de datos, database authorization, ult, luw, logic unit of work, shadow copy, acid, diseño, 1fn, fnbc, 2fn, 3fn, 4fn, fnrp, fndc, forma normal de dominio y clave, reflexividad, transitiva, descomposición, descomposición de esquemas, algoritmo de descomposición, canonical recover, repetición de la información, dinamic hash, static hash, indización, tree b+, b+ tree, hash index, hash, tabla hash, bucket, ordered index, secondary index, sparse index, dense index, primary key, key, search key | es_sinonimo_de |

Cuadro A.3: Tabla de instancias de Teoría de Bases de Datos.

# Apéndice B

# Diagramas de ontologías

## B.1. Referencia de clases para leer los diagramas, ver cuadro B.1

| Concepto Clase | Representación de diagrama |
|---|---|
| Área |   |
| Propiedad de área |   |
| SubÁrea |   |
| Componente |   |
| Sínonimo |   |

Cuadro B.1: Referencias de diagramas de los *Conceptos* de la ontología.

**B.2. Ontología de Agentes Inteligentes (AI), ver figura B.1**

**B.3. Ontología de Aprendizaje Automático (AA), ver figuras B.2 y B.3**

**B.4. Ontología de Teoría de Bases de Datos (TBD), ver figuras B.4, B.5, B.6 y B.7**

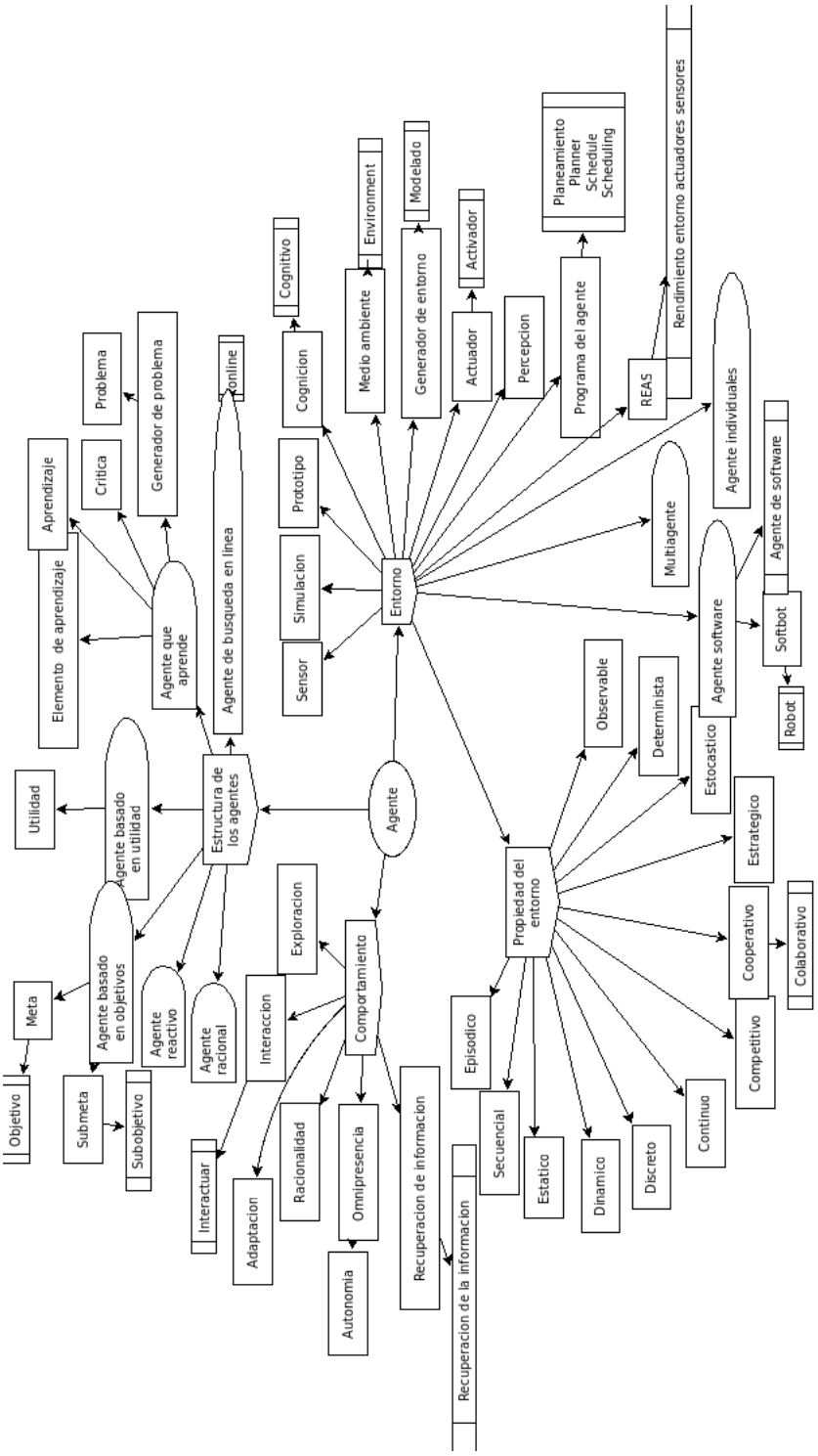



Figura B.1: Ontología de Agentes Inteligentes (AI).

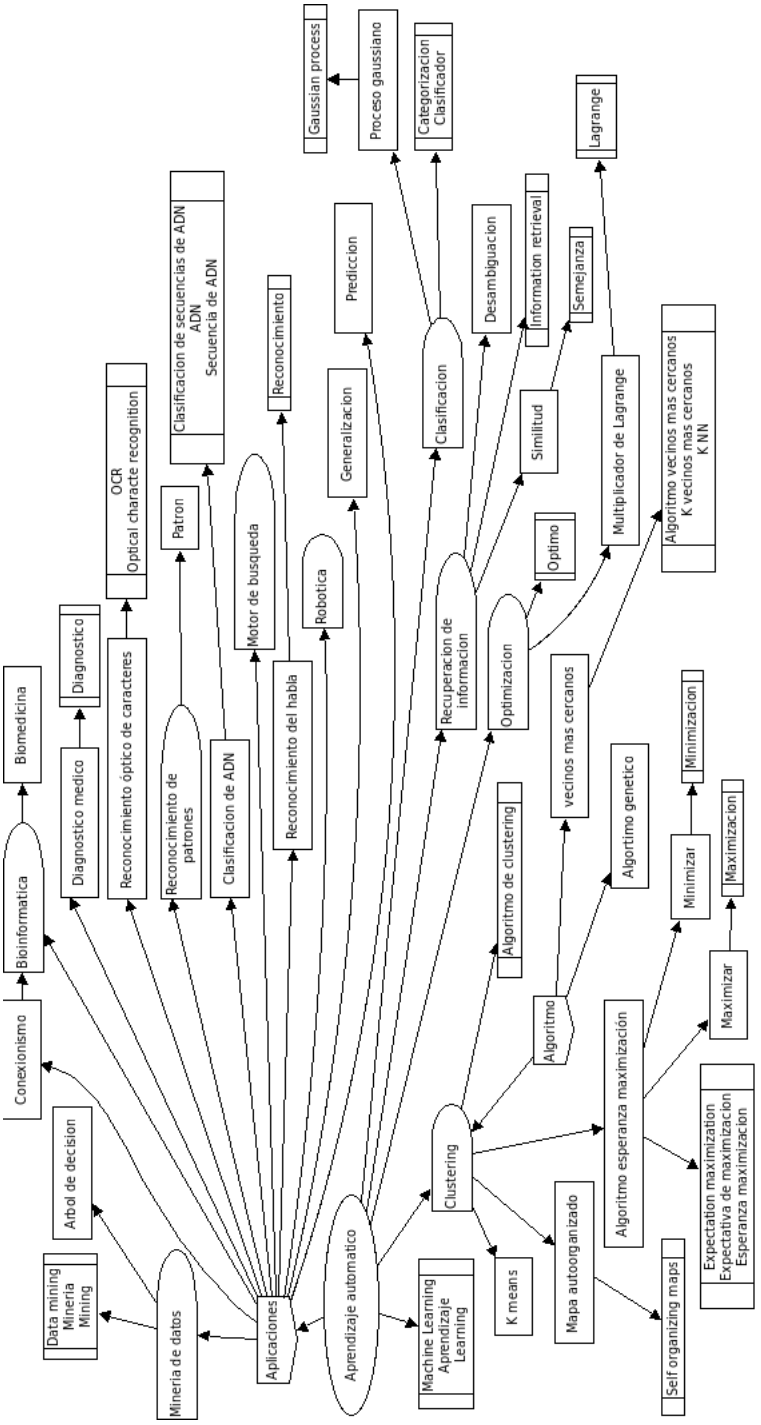



Figura B.2: Ontología de Aprendizaje Automático (AA), parte I.

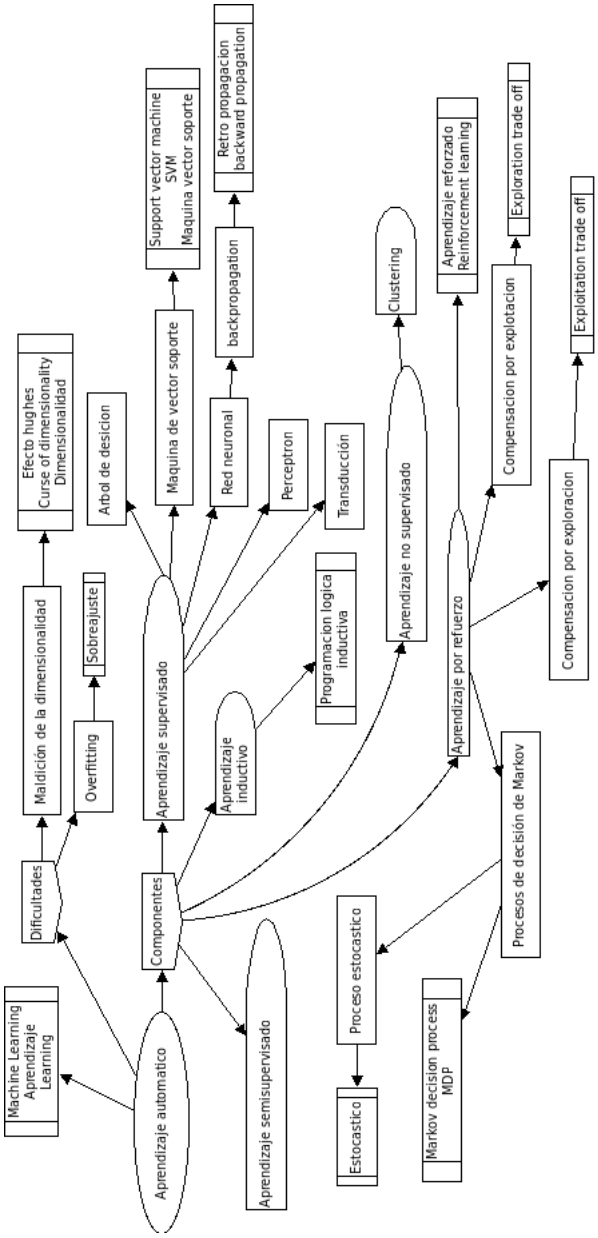



Figura B.3: Ontología de Aprendizaje Automático (AA), parte II.

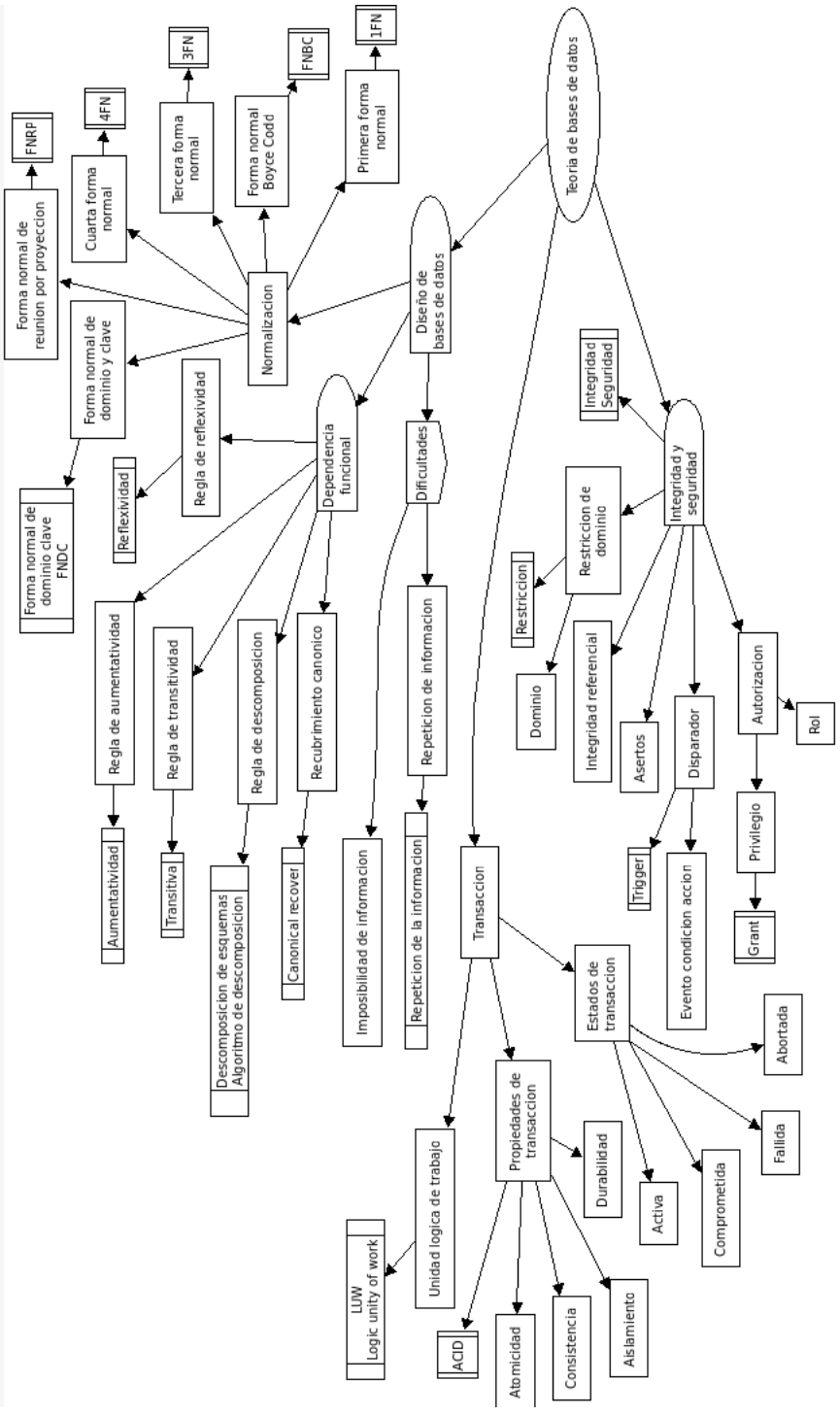



Figura B.4: Ontología de Teoría de Bases de Datos (TBD), parte I.

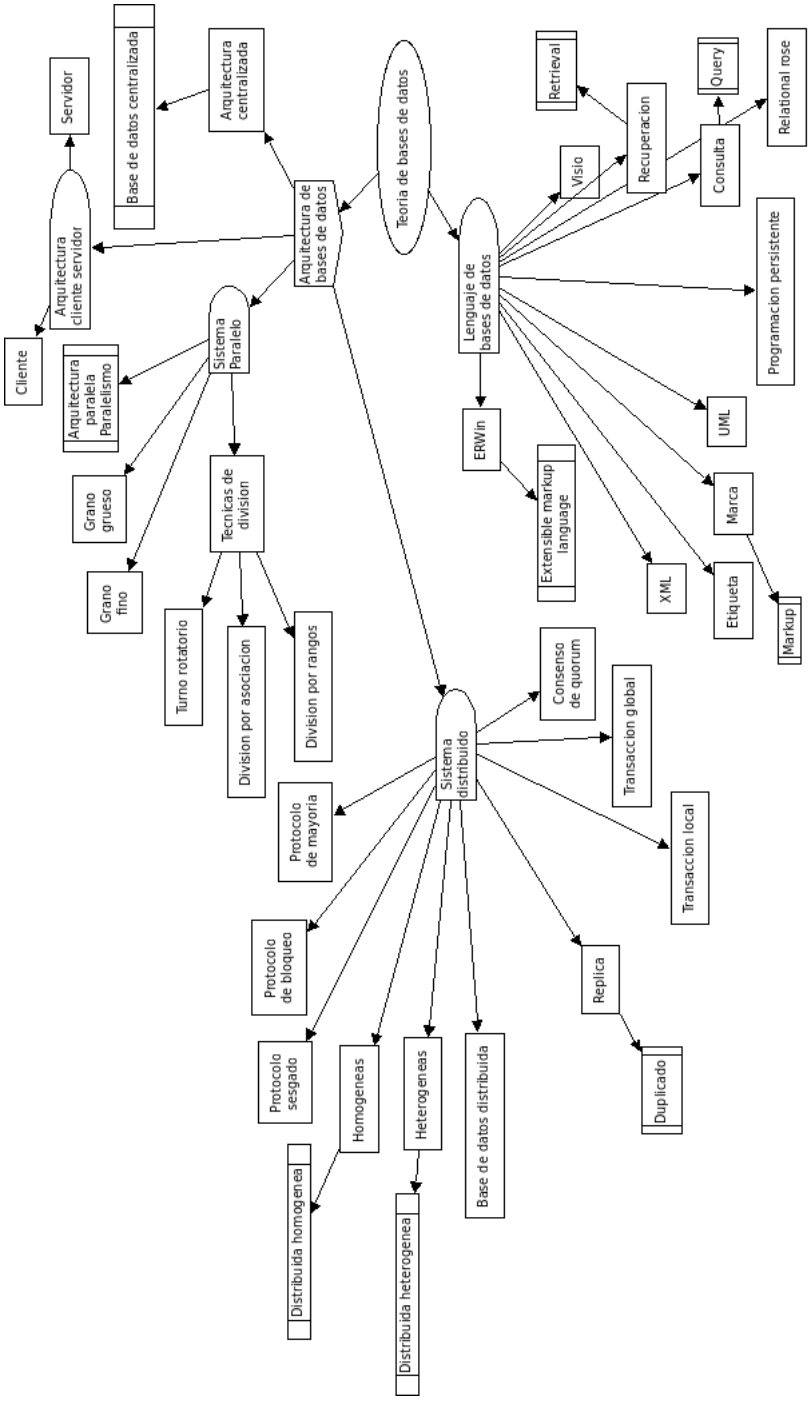



Figura B.5: Ontología de Teoría de Bases de Datos (TBD), parte II.

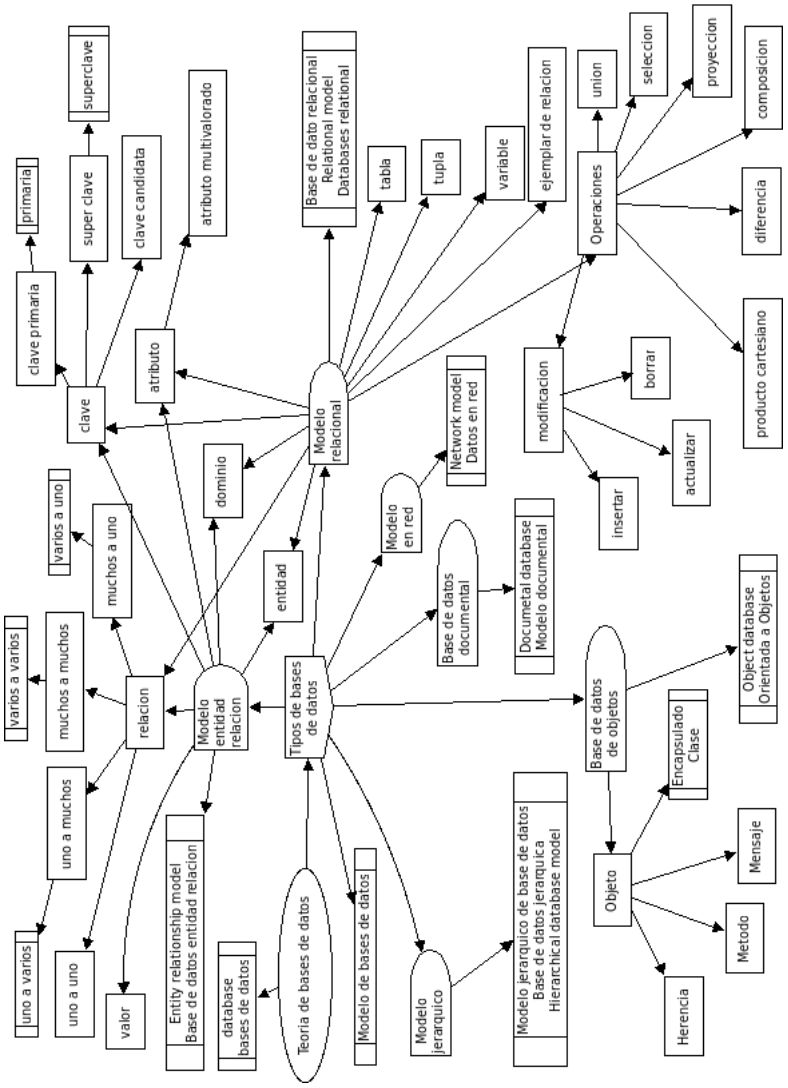



Figura B.6: Ontología de Teoría de Bases de Datos (TBD), parte III.

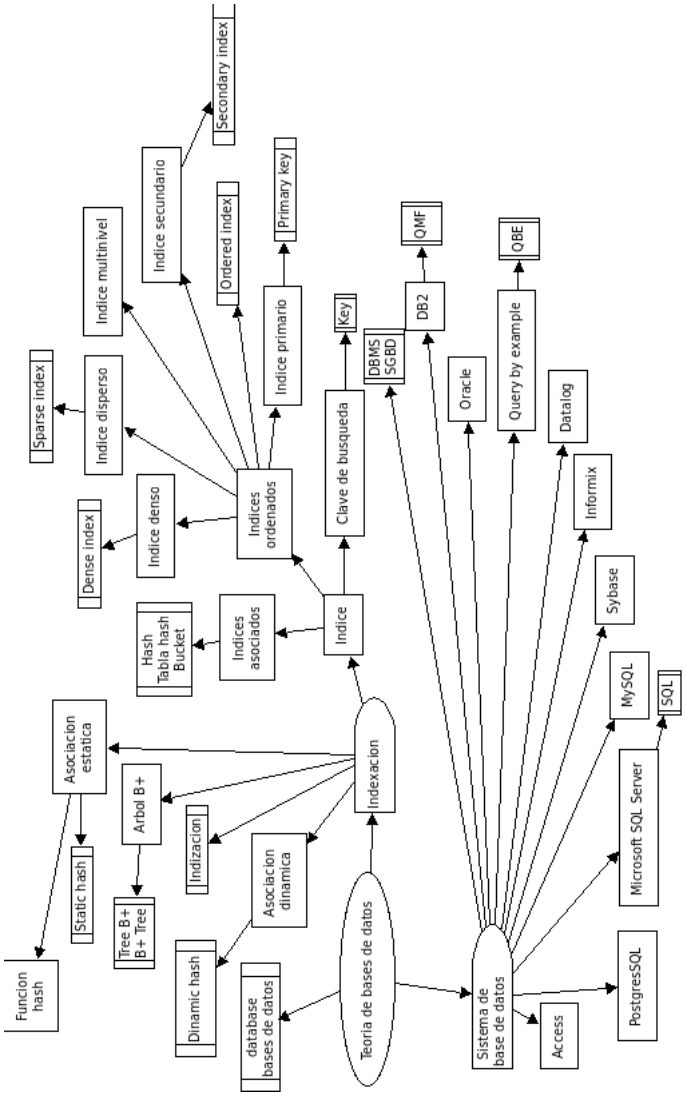



Figura B.7: Ontología de Teoría de Bases de Datos (TBD), parte IV.

# Bibliografía


[1] Al-Baltah, I. A., Ghani, A., Azim, A., Rahman, W. A., Nurhayati, W., and Atan, R. A comparative study on ontology development methodologies towards building semantic conflicts detection ontology for heterogeneous web services. *Research Journal of Applied Sciences, Engineering and Technology 7*, 13 (2014), 2674–2679.

[2] Anjewierden, A., and Kabel, S. Automatic indexing of pdf documents with ontologies. In *Proceedings of the 13th BelgiumNetherlands Conference on Artificial Intelligence (BNAIC 2001)* (2001), p. 23.

[3] Aranda, G. N., and Ruíz, F. Clasificación y ejemplos del uso de ontologías en ingeniería del software. *XI Congreso Argentino de Ciencias de la Computación* (2005). Departamento de Ciencias de la Computación, Universidad Nacional del Comahue. Departamento de Informática, Universidad de Castilla.

[4] Biebow, B., and Szulman, S. Terminae: a method and a tool to build a domain ontology.

[5] Borst, W. N. *Construction of engineering ontologies for knowledge sharing and reuse*. Universiteit Twente, 1997.

[6] Busser, R. D. *Information Extraction and Information Technology*. Springer Netherlands, 2006, pp. 1–22.

[7] Frank, E., and Medelyan, O. Keyword extraction algorithm. http://www.nzdl.org/Kea/index.html. Última visita: Septiembre 2015.

[8] García, R. V. *Un Entorno para la Extracción Incremental de Conocimiento desde Texto en Lenguaje Natural*. PhD thesis, Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, 2005.

[9] Gazendam, L., Wartena, C., and Brussee, R. Thesaurus based term ranking for keyword extraction. *Database and Expert Systems Applications (DEXA), 2010 Workshop on* (2010).

[10] Giaretta, P., and Guarino, N. Ontologies and knowledge bases towards a terminological clarification. *Towards very large knowledge bases: knowledge building & knowledge sharing 25* (1995), 32.

[11] Gómez-Pérez, A., Fernández, M., and Juristo, N. Methontology: From ontological art towards ontological engineering. Tech. rep., Laboratorio de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid., 2001.

[12] Gruber, T. Ontology. http://tomgruber.org/writing/ontology-definition-2007.htm. Última visita: Agosto 2016.

[13] Grüninger, M., and Fox, M. S. Methodology for the design and evaluation of ontologies. Tech. rep., Departamento de Ingeniería Industrial, Universidad de Toronto, 1995.

[14] Gutiérrez, J. B. N.-C. Catalogación y búsqueda semántica en un sitio web. *XXXI Conferencia Latinoamericana de Informática, Cali, Colombia.* (2005).

[15] IBM. Alchemyapi. http://www.alchemyapi.com. Última visita: Septiembre 2015.

[16] Lenat, D., and Guha, R. Building large knowledge-based systems: Representation and inference in the cyc project. *Artificial Intelligence 61*, 1 (1993), 41–52.

[17] Lexalytics. Semantria. https://semantria.com/support/resources/technology. Última visita: Septiembre 2015.

[18] Luna, J. A. G., Bonilla, M. L., and Torres, I. D. Metodologías y métodos para la construcción de ontologías. *Scientia et Technica 2*, 50 (2012), 133–140.

[19] Medelyan, O. Automatic keyphrase indexing with a domain-specific thesaurus. *Master's thesis, Albert-Ludwigs University* (2005).

[20] Mizoguchi, R., Vanwelkenhuysen, J., and Ikeda, M. Task ontology for reuse of problem solving knowledge. *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing* (1995), 46–59.

[21] Nejdl, W., Vollmer, H., and Demidova, E. Automatic keyword extraction for database search. Tech. rep., L3S Research Center, 2009.

[22] Noy, N. F., and Mcguinness, D. L. Ontology development 101: A guide to creating your first ontology. Tech. rep., Universidad de Stanford, 2001.

[23] Redonnet, J. P. Texlexan. http://www.sourceforge.net/projects/texlexan/files/. Última visita: Septiembre 2015.

[24] Redonnet, J. P. Texlexan. http://www.texlexan.sourceforge.net/. Última visita: Septiembre 2015.

[25] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. In *Information Processing And Management* (1988), pp. 513–523.

[26] Skuce, D. Conventiones for reaching agreement on shared ontologies. *Proceedings of the 9th knowledge acuisition for knowledge based systems workshop.* (2005).

[27] Sowa, J. Ontology. http://www.jfsowa.com/ontology/. Última visita: Agosto 2016.

[28] Uschold, M., and King, M. Towards a methodology for building ontologies. *Workshop on basic ontological issues in knowledge sharing.* (1995).

[29] Wijewickrema, P., and Gamage, R. Automatic document classification using a domain ontology. Tech. rep., Universidad de Colombo, Sri Lanka, 2012.

[30] Yedid, N. Modelos de indización temática utilizados en repositorios digitales de acceso abierto de argentina. Tech. rep., Universidad de Buenos Aires. Facultad de Filosofía y Letras. Departamento de Bibliotecología, 2013.