UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Centro Singular de Investigación en Tecnoloxías da Información

Tesis doctoral

DESCUBRIMIENTO DE GRAFOS EN DATOS ENLAZADOS PARA LA ANOTACIÓN SEMÁNTICA DE DOCUMENTOS

Presentada por:

Estefanía Natalia Otero García

Dirigida por:

Manuel Lama Penín

Juan Carlos Vidal Aguiar

Febrero de 2017

Manuel Lama Penín, Profesor Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Santiago de Compostela

Juan Carlos Vidal Aguiar, Profesor Asociado de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Santiago de Compostela

HACEN CONSTAR:

Que la memoria titulada **DESCUBRIMIENTO DE GRAFOS EN DATOS ENLAZADOS PARA LA ANOTACIÓN SEMÁNTICA DE DOCUMENTOS** ha sido realizada por **Estefanía Natalia Otero García** bajo nuestra dirección en el Centro Singular de Investigación en Tecnoloxías da Información de la Universidad de Santiago de Compostela, y constituye la Tesis que presenta para optar al título de Doctor.

Febrero de 2017

Firma
Manuel Lama Penín

Firma
Juan Carlos Vidal Aguiar

Firma
Estefanía Natalia Otero García

Mis mayores éxitos fueron producto de decisiones que tomé cuando dejé de pensar e hice sencillamente lo que me parecía correcto.

Kvothe

La ciencia se construye a partir de aproximaciones que gradualmente se acercan a la verdad.

Isaac Asimov

Agradecimientos

En estas líneas quiero agradecer a todas aquellas personas que, de una u otra forma, han contribuido a que esta tesis fuera posible:

A mis directores, Manuel Lama y Juan Carlos Vidal, por la confianza que han depositado en mí para llevar a cabo esta tesis. Gracias a la oportunidad que me dieron, he podido experimentar lo que significa ser una investigadora, ayudándome a crecer como profesional y como persona.

Al Departamento de Electrónica y Computación y al Centro Singular de Investigación en Tecnologías de la Información (CiTIUS), por haber podido llevar a cabo mi investigación en estos centros. También una mención a sus investigadores y personal de apoyo, en especial a Paulo Félix, por haberme ofrecido sus consejos, que mucho me han ayudado.

Un gracias gigantesco a mis compañeros y amigos del CiTIUS: Adri, Ana, Ángel, Borja, David, Ismael, Jano, Jorge, Pablo, Tino, Tomás y Víctor, que me han apoyado en este largo viaje. Han convertido este camino en uno mucho más divertido de recorrer.

A mis amigas Elena y Laura, por convertir el lado oscuro de las cosas en ironía y sátira con conversaciones que me sacan muchas sonrisas. También a Vero y Fran, por los días de turismo junto a Ana y Adri.

A mi familia, mis padres, Jorge y Alicia, y mi hermana Johanna, por el infinito cariño que me han dado siempre. Sin ellos nunca habría llegado hasta aquí. A Luly, que sigue en el recuerdo. A las familias Casal y Villar, a las que prácticamente siento como mías. A Fari, que

todavía sigue en pie. Y finalmente, aunque no menos importante, a Arturo, por quererme, apoyarme y por alentarme a que continuara a pesar de algunas piedras que me han ido aparecido por el camino.

Febrero de 2017

Índice general

Prefacio

La cantidad de datos digitales disponibles en los últimos años ha aumentado exponencialmente, convirtiendo a la Web en el mayor repositorio de información, donde gran parte de los datos se encuentran en texto plano y expresados en lenguaje natural. La recuperación de información (RI) proporciona herramientas que facilitan la búsqueda de estos datos de naturaleza no estructurada satisfaciendo unas necesidades de información dentro de una colección de documentos. Sin embargo, la mayoría de las técnicas de RI están basadas en una aproximación sintáctica que limita el acceso a la información relevante por parte de los usuarios, sobre todo cuando el volumen de información es muy significativo. La Web Semántica [7] trata de resolver en parte este problema, dotando de mayor significado al conocimiento disponible en la Web, permitiendo a las máquinas entender y compartir esta información de una forma más sencilla. En este contexto, han surgido los *datos enlazados* (LD, del inglés *Linked Data*) [8], como una aproximación para vincular contenidos de fuentes heterogéneas, y proporcionar una semántica y un significado a través de los enlaces que los relacionan. De esta forma, el contenido pasa de ser textual a ser semántico, con un significado concreto. La aparición de los datos enlazados, además, ha contribuido a que la investigación en anotación semántica haya experimentado una segunda juventud, en la que se hace uso de los extensos repositorios que los contienen, como fuente para anotar y clasificar documentos.

Contexto de investigación

Podemos entender el proceso de anotación de documentos como una forma de adjuntar ciertos metadatos como frases, comentarios o etiquetas a un documento o a parte de él [51]. La anotación semántica extiende este concepto y va un paso más allá para reducir el espacio entre el lenguaje natural y su representación computacional, tratando de emparejar los

términos de un documento con su representación semántica en una ontología, encargada de representar el conocimiento de forma estructurada [53]. Aun así, el mismo término puede tener múltiples significados o incluso términos distintos pueden hacer referencia al mismo concepto. Pongamos como ejemplo el término *Paris* en un documento, donde puede hacer referencia a la capital de Francia, un personaje mitológico griego o un nombre de persona en la cultura anglosajona, entre otras entidades. Encontrar en este caso su representación correcta en una ontología es esencial para proporcionar una anotación precisa. Para conseguir este propósito, los sistemas de anotación explotan el *contexto* de un término o documento, es decir, los términos que aparecen en el texto y que proporcionan significado al documento. Utilizando el contexto en el proceso de anotación se aumenta la precisión a la hora de buscar las entidades asociadas a los términos a anotar. Por ejemplo, si el término *Francia* o *capital de Francia* apareciese en el texto, la identificación del significado correcto sería mucho más sencilla al poder descartar otras entidades que no están relacionadas con dichos términos. La ventaja de usar una ontología en el proceso de anotación es que el modelo se define en la etapa de conocimiento [79], permitiendo evaluar las relaciones entre los conceptos o entidades para reducir la incertidumbre a la hora de seleccionar el significado correcto. Siguiendo el mismo ejemplo, en el caso de *Paris*, en la ontología debería existir una relación que indicaría que es la capital de *Francia*.

En este contexto se plantea el uso de repositorios de datos enlazados en la anotación semántica. Este uso implica, por una parte, una mejora en los resultados de la anotación, debido a que la cantidad de entidades y relaciones entre ellas es enorme, proporcionando un mayor conocimiento a la hora de seleccionar la entidad correcta; y por otra parte, un menor rendimiento debido a la necesidad de procesar o evaluar una mayor cantidad de información. Por lo tanto la identificación de la entidad correcta ya no se basa sólo en el contexto, sino en una exploración de este grafo de conocimiento para desambiguar entre las posibles entidades que podrían representar la anotación de un término.

Nuestro trabajo de investigación se contextualiza dentro del enriquecimiento de la información no estructurada mediante enlaces hacia repositorios de conocimiento estructurado. Lo que formalmente se conoce en anotación semántica como *enlazado de entidades* (*EL*, del inglés *Entity Linking*). Este trabajo permite analizar textos expresados en lenguaje natural para identificar un conjunto de términos clave en el texto y añadir metadatos, entendidos como relaciones, a entidades de un repositorio de datos enlazados, que permitan generar una estructura semánticamente sencilla de comprender por un ordenador. De esta forma, se mejoran

procesos de recuperación y clasificación del texto, se presenta el texto generado por las personas de una forma comprensible a las máquinas y, lo más importante, se enriquece el texto con enlaces hacia las entidades que representan dichos conceptos.

Hipótesis

El objetivo de los sistemas de anotación semántica actuales es enlazar los términos relevantes de un documento con entidades que los representan dentro de un repositorio semántico de conocimiento. Sin embargo, aunque los repositorios de datos enlazados representan el conocimiento a través de grafos semánticos, por lo general los sistemas de anotación no realizan una exploración de dichos grafos con el fin de realizar una búsqueda contextual de las entidades solución. En otras palabras, no explotan el grafo semántico de relaciones a la hora de resolver el problema de anotación. Teniendo esto en cuenta, en esta tesis la hipótesis de partida es la siguiente:

"El uso de técnicas de exploración y filtrado semántico de grafos en repositorios de datos enlazados mejora los resultados de la anotación y facilita el enriquecimiento de la información al relacionar cada término con un grafo semántico."

Como resultado de la anotación con repositorios de datos enlazados, para cada término relevante del documento se obtendrá un grafo semántico cuyos nodos o entidades están pesados con una relevancia que indica su grado de relación con el término del documento.

Objetivo

El objetivo de esta tesis doctoral consiste en el desarrollo de un sistema de anotación semántica basado en contexto para la anotación de documentos con grafos semánticos extraídos de repositorios de datos enlazados. Para alcanzar este objetivo general, se proponen los siguientes objetivos específicos:

O1. Desarrollo de un sistema de exploración y filtrado de grafos semánticos asociados a los repositorios de datos enlazados para la extracción de subgrafos con los que se anotan los términos relevantes del documento.

O2. Aplicación del sistema de anotación semántica para resolver problemas reales de extracción de palabras clave y clasificación de documentos.

O3. Validación del sistema de anotación y comparación con otras propuestas del estado del arte.

Contribuciones de la tesis

Teniendo en cuenta los objetivos previos, las principales contribuciones de la tesis son las siguientes:

C1. Desarrollo de un algoritmo de filtrado de grafos semánticos que obtiene los subgrafos con los que se anotan los términos relevantes inicialmente identificados en el documento. La construcción de estos grafos nos permiten relacionar entre sí los conceptos asociados a los términos, eliminando aquellos que no están relacionados con el documento de partida y mejorando con ello la identificación de los términos relevantes. También nos permite realizar una mejor desambiguación de los conceptos del repositorio que intentamos enlazar a los términos relevantes del documento, ya que las relaciones entre los conceptos del repositorio nos ayudan a contextualizar los términos y mejorar la identificación de los conceptos más adecuados.

C2. Desarrollo de un algoritmo de exploración del repositorio de datos enlazados que es parametrizable en profundidad, no limitada por el propio algoritmo. Además, esta exploración es independiente del tipo de relaciones y de la estructura interna del repositorio, pudiendo aplicarse a cualquier repositorio que cubra un dominio específico o sea de propósito general.

C3. Creación de un grafo de conceptos semánticamente relacionados, obtenidos como resultado de los algoritmos de exploración y filtrado de grafos, que nos permite enriquecer y extender el contenido de los documentos así como mejorar procesos de búsqueda y clasificación.

C4. Comparación del sistema de anotación con otras propuestas del estado del arte, usando los conjuntos de datos de validación públicamente disponibles y aceptados por la comunidad investigadora.

C5. Aplicación del sistema de anotación semántica en un problema real y, más concretamente, en la clasificación de objetos de aprendizaje de una biblioteca virtual.

Estructura de la memoria

La memoria de la tesis está estructurada en seis capítulos, que incluyen el estado del arte, la descripción del sistema de anotación y del experimento de aplicación real, y las conclusiones. La estructura es la siguiente:

- En el primer capítulo, ofrecemos una revisión de las principales propuestas que tratan de abordar el problema de la anotación semántica desde diferentes perspectivas. Indicamos las carencias que existen en el estado del arte y la falta de soluciones que hay relacionadas con la anotación semántica y el enriquecimiento de información a través de datos enlazados.

- En el segundo capítulo presentamos una versión inicial de un sistema de anotación semántica, denominado *ADEGA*, cuyo principal objetivo es representar el contenido relevante de un texto mediante un grafo extraído de la DBpedia que proporcione conceptos relacionados a partir de los enlaces identificados para los términos importantes, de forma que la información se vea enriquecida como resultado del proceso de anotación.

- En el tercer capítulo presentamos una solución mejorada de *ADEGA* para abordar el problema de rendimiento en la exploración de los grafos solución debido al uso de un algoritmo de búsqueda en profundidad. Como solución, proponemos la paralelización del algoritmo utilizando una aproximación basada en la integración de un grid, un clúster y recursos computacionales en la nube para afrontar la tarea de anotación en un tiempo razonable.

- En el cuarto capítulo proponemos una nueva versión de *ADEGA* para la mejora del rendimiento, ya que no es asumible disponer de una arquitectura de integración de grandes cantidades de recursos computacionales para conseguir una anotación eficiente. Se da solución a los problemas existentes de la primera versión del algoritmo, y se ofrece una comparativa del sistema con los algoritmos de anotación del estado del arte.

- En el quinto capítulo describimos un caso práctico de aplicación de *ADEGA* para la generación de metadatos en un conjunto de objetos de aprendizaje del repositorio de Universia. El objetivo es clasificar los objetos de aprendizaje con un conjunto de categorías extraídas automáticamente de la DBpedia, además de proporcionar metadatos adicionales para enriquecer el contenido de los objetos.

- En el sexto y último capítulo se exponen las conclusiones de la investigación realizada, resumiendo las aportaciones más importantes del trabajo.

Publicaciones

Las contribuciones de esta tesis están incluidas en las publicaciones que se detallan a continuación.

Revistas internacionales

- Manuel Lama, Juan C. Vidal, Estefanía Otero-García, Alberto Bugarín y Senén Barro. *Semantic linking of learning object repositories to DBpedia*. Educational Tecnhology & Society, 15(4): 47-61, 2012. JCR 1,171, *Q1*.

- Juan C. Vidal, Manuel Lama, Estefanía Otero-García y Alberto Bugarín. *Graph-based Semantic Annotation for enriching educational content with Linked Data*. Knowledge-Based Systems, 55: 29-42, 2014. JCR 2,947, *Q1*.

- Javier Fabra, Sergio Hernández, Estefanía Otero-García, Juan C. Vidal, Manuel Lama y Pedro Álvarez. *Integration of Grid, cluster and cloud resources to semantically annotate a large-sized repository of learning objects*. Concurrency and Computation: Practice and Experience, 27(17): 4603-4629, 2015. JCR 0,942, *Q3*.

Revistas nacionales

- Sergio Hernández, Estefanía Otero-García, Juan C. Vidal, Javier Fabra, Manuel Lama y Pedro Álvarez. *Integración de recursos grid y cloud para anotar semánticamente grandes colecciones de objetos de aprendizaje*. Novática, 225: 56-61, 2014.

Congresos

- Estefanía Otero-García, Juan C. Vidal, Manuel Lama, Alberto Bugarín y José E. Domenech. *A context-based algorithm for annotating educational content with Linked Data*. 3rd Future Internet Symposium (FIS 2010), FIS-Workshop on Mining Future Internet (MIFI 2010), 32-37.

- Estefanía Otero-García, Juan C. Vidal, Manuel Lama, Alberto Bugarín y José E. Domenech. *Semantic Annotation of educational resources through Linked Data*. International Conference on Web-Based Learning (ICWL 2010 Workshops), 311-320, Springer Berlin Heidelberg, 2010.

- Estefanía Otero-García, Juan C.Vidal, Manuel Lama, Alberto Bugarín y José E. Domenech. *Toward enriching course content with Linked Data*. World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA 2011), 6: 3887-3894, 2011.

- Manuel Lama, Juan C. Vidal, Estefanía Otero-García, Alberto Bugarín y Senén Barro. *Semantic linking of a learning object repository to DBpedia*. 11th IEEE International Conference on Advanced Learning Technologies (ICALT-2011), 460-464, IEEE, 2011.

- Juan C. Vidal, Manuel Lama, Estefanía Otero-García y Alberto Bugarín. *An Evolutionary Approach for Learning the Weight of Relations in Linked Data*. 11th International Conference on Intelligent Systems Design and Applications (ISDA-2011), 1002-1007, IEEE, 2011.

- Javier Fabra, Sergio Hernández, Pedro Álvarez, Estefanía Otero-García, Juan C. Vidal y Manuel Lama. *A practical experience concerning the parallel Semantic Annotation of a large-scale data collection*. 9th International Conference on Semantic Systems (I-SEMANTICS 2013), 61-72, ACM, 2013.

- Sergio Hernández, Estefanía Otero-García, Javier Fabra, Juan C. Vidal, Manuel Lama y Pedro Álvarez. *Una experiencia real de anotación semántica a gran escala utilizando recursos de computación heterogéneos*. IX Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2013), 163-177, 2013.

Capítulo 1

Anotación semántica

1.1. Introducción

El campo de la anotación semántica ha ganado mucha atención en el ámbito investigador, influyendo positivamente en la evolución del problema que intentaban solucionar. Inicialmente, los sistemas de anotación disponibles centraron los esfuerzos en resolver problemas de *reconocimiento de nombres de entidades* (*NER*, del inglés *Named Entity Recognition*) [36], una subtarea del campo de *Extracción de Información* que, utilizando técnicas de procesamiento de lenguaje natural, intenta identificar términos en un texto y clasificarlos en categorías predefinidas como nombres de personas, localizaciones, valores monetarios, expresiones temporales o porcentajes. Con el creciente desarrollo de grandes repositorios como DBpedia, Freebase, o tesauros como YAGO, estas tareas han ido tornándose más sofisticadas. En estos repositorios, donde el conocimiento está modelado de forma semántica, los datos representan grandes cantidades de conceptos enlazados entre sí; y estos enlaces modelan relaciones con un significado específico. El volumen de información es considerable; el mismo concepto puede modelar entidades de la realidad totalmente distintas que dependiendo del significado, puede contener un conjunto de relaciones u otro hacia el resto de conceptos. En un escenario como este, las tecnologías evolucionaron tratando de enlazar los términos del texto a sus correspondientes representaciones en un repositorio. La tarea pasó a denominarse *desambiguación de nombres de entidades* (*NED*, del inglés *Named Entity Disambiguation*) [75], ya que era necesario realizar un proceso de identificación del concepto concreto que modela el significado de un término dentro de un contexto, eliminando en este caso la polisemia. Este problema es la evolución de uno más genérico que ya existía antes de la creación de estos repositorios:

desambiguación lingüística de términos (*WSD*, del inglés *Word Sense Disambiguation*), un proceso donde se identifica el significado concreto de un término usado en una frase, en el caso de que el término tuviera múltiples acepciones. De esta forma han nacido las tareas de anotación semántica más genéricas denominadas:

- *Enlazado de Entidades* (*EL*, del inglés *Entity Linking*) [94], donde se establecen enlaces a repositorios semánticos entre los términos más relevantes del texto y los conceptos que los representan.

- *Enlazado de Nombres de Entidades* (*NEL*, del inglés *Named Entity Linking*) [12], refinamiento de la tarea anterior, donde se anota un conjunto específico de términos: aquellos que correspondan con nombres de personas, localizaciones, etc.

Las propuestas que resuelven estos problemas, han basado sus soluciones en la utilización de un conjunto de características léxicas y semánticas obtenidas de los repositorios semánticos con los que se establecía la anotación. Mayoritariamente, estas características en un primer momento, estaban obtenidas de repositorios como Wikipedia, pero posteriormente de ontologías como DBpedia, donde se introduce más contenido semántico que léxico.

Aunque hay muchas propuestas que resuelven el problema de *EL* o *NEL* de forma más o menos satisfactoria, ninguna de ellas utiliza el proceso de anotación como medio para enriquecer la información de partida. Creemos que identificar metadatos en relación a un texto para obtener una representación donde las máquinas entiendan el significado es imprescindible, pero quedarse en la representación de los conceptos mediante un único enlace a un repositorio de conocimiento es insuficiente para dar solución a un proceso de enriquecimiento de la información. Nuestra propuesta trata de resolver el problema de *EL* extendiendo el proceso de anotación semántica, de forma que cada concepto no solo se represente mediante un enlace a un concepto concreto, sino que a partir de él se expanda utilizando un grafo semántico que contenga conceptos relacionados al contenido de partida. El resultado de la anotación de un término no es un solo concepto, sino un grafo semántico.

En las siguientes secciones analizamos las propuestas que existen en el estado del arte de acuerdo a tres tareas que consideramos importantes en el proceso de enlazado de entidades para saber cómo han abordado el problema en cada una de ellas, cuál es el resultado de la anotación que realizan sobre un término y para qué fin utilizan esta anotación una vez se realiza.

1.2. Enlazado de entidades

Consideramos que existen tres tareas básicas para realizar *EL* [67]: la identificación de términos en el texto, la selección de las entidades candidatas del repositorio para realizar la anotación; y la desambiguación de las entidades para identificar aquella que representa al término. Para poder entender cada una de estas tareas es necesario explicar la siguiente nomenclatura que se usará a lo largo de la memoria:

- Un *término*, t, es una palabra o conjunto de palabras localizadas en el texto de entrada.
- Una *mención*, m, es la ocurrencia de un término en el texto y puede codificarse por el par (p,l). p es la posición de la mención dentro del texto de partida y l es la longitud de la misma.
- Una *entidad*, e, o concepto es aquel recurso del repositorio que representa al término, identificado unívocamente por una URI.
- Una *anotación* es el enlazado de una mención de un término a una entidad y es codificado por el par (m,e).

Cuando hablamos de *identificación de términos*, hacemos referencia al conjunto de términos relevantes e importantes de un documento. Esta tarea también se ha identificado en el estado del arte como "extracción o detección de términos clave" [68, 71], "localización" (del inglés *spotting*) [67], o "contexto" [83]. La tarea de *selección de entidades candidatas* conlleva seleccionar todas las potenciales entidades del repositorio que pueden representar al término con el que pretendemos establecer la anotación. Por otro último, entendemos por *desambiguación de entidades*, *mapeado* [25] o *ranking* [77], a la acción de realizar un proceso de selección del mejor concepto del repositorio que representa al término de entre todos los candidatos generados en el paso previo. El ranking de los candidatos a desambiguar es un paso final en el proceso de selección del concepto correcto, ya que el proceso de desambiguación en sí mismo asigna una calificación a cada uno de los candidatos, produciendo como salida un conjunto de candidatos ordenados según la puntuación o grado de confianza de ser la entidad correcta.

Algunos trabajos propuestos también intentan mapear aquellos términos que no tienen una representación en el repositorio semántico utilizado [88], mayoritariamente Wikipedia y DBpedia. Dado que la mayor parte de los conjuntos de datos utilizados para evaluar estos

sistemas no contemplan esta posibilidad, esta característica no se tendrá en cuenta durante la revisión de las propuestas.

1.2.1. Repositorios semánticos para anotación

La mayor parte de las propuestas utilizan como repositorios de conocimiento en el proceso de anotación semántica a la Wikipedia, WordNet o DBpedia. Por este motivo creemos necesario proporcionar un pequeño resumen de estos recursos, en aras de una mejor comprensión de las características que se obtienen de ellos para llevar a cabo las tareas anteriormente descritas.

1.2.2. Wikipedia

La Wikipedia, según su propia definición, es una enciclopedia web libre, políglota y editada colaborativamente, donde los voluntarios proporcionan un repositorio de conocimiento enciclopédico de gran cobertura. De las 284 ediciones en diferentes idiomas, 28 superan los 300.000 artículos, mientras que la versión en inglés contiene más de 5 millones.

Cada artículo de la Wikipedia, que representa una página, presenta información específica sobre un concepto (por ejemplo, "Biblioteca") o entidad (en relación a personas, localizaciones, valores monetarios, expresiones temporales o porcentajes; por ejemplo, "Miguel de Cervantes"). El contenido de la Wikipedia está parcialmente estructurado. Además del contenido textual expresado en texto plano, las páginas contienen *cajas de información*, un tipo especial de tabla que resume los atributos más importantes del concepto o entidad. Suelen estar localizadas a la derecha de la página. Cada caja de información se construye a partir de una plantilla que contiene propiedades específicas según el dominio. Por ejemplo, en el caso de consultar una página de una persona, encontraríamos información como edad, fecha y lugar de nacimiento; en el caso de consultar una localización, podríamos obtener información como país al que pertenece, ubicación geográfica, población, etc. Adicionalmente, existen ciertas relaciones entre las páginas:

- **Páginas de redirección**: cada artículo contiene un conjunto de páginas de redirección que enlazan al mismo artículo. Estas páginas suelen expresar una forma alternativa de referirse al mismo concepto dentro del repositorio: son sinónimos del nombre del artículo, acrónimos o incluso errores ortográficos comunes. Por ejemplo, el artículo *"Leonardo da Vinci"* redirige desde *Leonardo*.

- **Páginas de desambiguación**: tratan de modelar la polisemia de los conceptos, haciendo referencia a otras páginas que modelan significados diferentes para el mismo término. Por ejemplo, el término *"París"*, que puede hacer referencia al concepto *París, capital de Francia* o *París, película del 2008*.

- **Enlaces internos**: todo artículo de la Wikipedia contiene enlaces internos dentro del texto hacia otros artículos, que normalmente mantienen cierta relación con el artículo principal. Por ejemplo, el artículo sobre *"Santiago de Compostela"* contiene enlaces internos a *España*, *A Coruña*, *Patrimonio de la Humanidad* o *Jerusalén*, por citar algunos.

- **Categorías**: los artículos pueden asignarse a una o más categorías, de forma que agrupen a los artículos que pertenecen a la misma temática. Existen subcategorías de una categoría. Esta característica permite organizarlas en una jerarquía con una estructura de árbol que facilita la navegación a través de ellas. Por ejemplo, el artículo de la banda de rock *"The Rolling Stones"* puede categorizarse en *cuartetos musicales*, *grupos de rock de Inglaterra*, o *grupos de música formados en 1962* entre otros.

1.2.3. WordNet

WordNet [31] es la base de conocimiento léxico más popular en el campo del procesamiento del lenguaje natural, utilizada con frecuencia en la tarea de desambiguación de palabras. Es una combinación de diccionario y tesauro para el inglés. Agrupa las palabras en conjuntos de sinónimos llamados *synsets*, proporcionando definiciones cortas y generales denominadas *gloss*. La base de datos contiene 155.287 palabras organizadas en 117.659 conjuntos de sinónimos.

WordNet distingue entre sustantivos, verbos, adjetivos y adverbios. Adicionalmente incluye relaciones entre conjuntos de sinónimos, que dependiendo de la morfología de los mismos, contiene unas relaciones u otras. Por ejemplo, para los sustantivos existen relaciones de hiperonimia (generalización), hiponimia (especificidad), holonimia y meronimia (relación semántica no simétrica que especifica el todo y la parte, respectivamente). Mientras que entre los verbos podemos encontrar relaciones como hiperonimia, troponimia (parte del significado de un verbo contenido en otro), consecuencia lógica (verbo consecuencia de otro, por ejemplo *dormir* es una consecuencia lógica de *roncar*) o términos coordinados (si comparten el mismo hiperónimo).

1.2.4. DBpedia

DBpedia [6] es un esfuerzo de la comunidad para extraer información estructurada de la Wikipedia y hacerla disponible en la Web [9]. Esta información se obtiene a través de un sistema de extracción automática de datos estructurados, con el objetivo de representar los conceptos de la Wikipedia y las relaciones entre ellos en una ontología en forma de tripletas RDF [55]. Una tripleta RDF consiste en un sujeto, un predicado y un objeto, donde normalmente el sujeto y el objeto hacen referencia a conceptos, mientras que el predicado es la relación semántica entre ellos. Cada uno se identifica a través de la Web por una URI (*Uniform Resource Identifier*). Gracias a esta representación, la DBpedia permite consultar semánticamente las relaciones y propiedades asociadas a los recursos de la Wikipedia a través de un lenguaje de consulta denominado SPARQL [86].

La DBpedia tiene ciertas ventajas en comparación con la representación de la información de la Wikipedia. Tiene un acceso a la información mucho más conveniente, además de estar estructurada; también proporciona calidad en los datos [60], esto es importante ya que la efectividad de un anotador depende directamente de la base de conocimiento seleccionada.

Las estadísticas obtenidas de la DBpedia en inglés a fecha de abril de 2016 indican que la ontología describe 4,6 millones de conceptos, donde más del 90 % está clasificado en una ontología consistente que incluye 1.445.000 personas, 735.000 lugares (de los cuales 478.000 son lugares poblados), 411.000 trabajos creativos (como 123.000 álbumes de música, 87.000 películas y 19.000 vídeo juegos), 241.000 organizaciones (correspondientes a 58.000 compañías y 49.000 instituciones educativas), 251.000 especies y 6.000 enfermedades.

El contenido de la DBpedia está organizado en diferentes conjuntos de datos en los que se representan los diferentes tipos de relaciones entre los conceptos. Cada entidad de la DBpedia contiene ciertas propiedades que la describen en cada conjunto:

- **Información textual básica**: donde se asocia a la entidad un nombre que se modela a través de una *etiqueta*, representada por la relación `rdfs:label`. También se le asocia una descripción corta o larga denominada *resumen* mediante la relación `dbo:abstract`.

- **Clasificación**: la DBpedia proporciona un sistema de clasificación de las entidades basado en una ontología estructurada para la clasificación de conceptos y la recuperación de los metadatos, el *vocabulario SKOS* [70] y los *términos DCMI* [49]. Las relaciones modelan la clasificación de las entidades en categorías, además de las relaciones de

jerarquía entre ellas. La DBpedia incluye un conjunto independiente de clasificación proporcionado por el sistema *YAGO* [97], creado a partir de las categorías de la Wikipedia usando WordNet. Las principales relaciones con las que se clasifican las entidades son las siguientes:

- `dct:subject`: esta relación enlaza a las categorías de una entidad. Las categorías son una forma alternativa de agrupar a las entidades, útil a la hora de recuperar aquellas con la misma clasificación. Por ejemplo, la entidad *dbr:España* pertenece a las categorías *dct:Países_en_Europa* o *dct:Estados_miembros_de_la_Unión_Europea*.
- `skos:broader`: esta propiedad establece una relación de jerarquía entre dos categorías. Concretamente, indica que una categoría es más general que otra. Por ejemplo, la categoría general de *dct:Países_en_Europa* es *dct:Europa*.
- `rdfs:type`: esta relación se usa para indicar que una entidad pertenece a una clase de la ontología, por ejemplo *dbr:España* pertenece a la clase de la ontología de la DBpedia *dbo:Lugar* y *dbo:País* entre otras. A través de esta relación también se incluyen las categorías de YAGO, que se pueden entender como la misma representación que ofrece la relación *dct:subject*. Siguiendo el mismo ejemplo, para la entidad *dbr:España* tendríamos la siguiente relación de tipología: *yago:WikicatCountriesInEurope*.
- `rdfs:subClassOf`: esta propiedad se usa para indicar que todas las instancias de una clase son instancias de una superior más genérica. Por ejemplo, la clase *dbo:País* es una subclase de la clase superior: *dbo:Lugar_Poblado*.

Para cada una de estas relaciones existe su inversa, por lo tanto tendríamos `is dct:subject of`, `is skos:broader of`, `is rdfs:type of` e `is rdfs:subClassOf of`.

- **Propiedades de las cajas de información**: todas las propiedades específicas generadas a partir de las plantillas de las cajas de información de la Wikipedia, se mapean a relaciones concretas para la entidad. Por ejemplo, *dbo:Ana_María_Matute* tiene la relación *dbprop:ocupación dbo:Escritora*. Cada una de las plantillas que conforman estas cajas se traducen en un conjunto específico de relaciones para un concepto.

- **Redirecciones**: la DBpedia contiene la relación `dbo:wikiPageRedirects` donde se modela exactamente el mismo comportamiento entre entidades que en la Wikipedia entre artículos. Esta relación apunta a un conjunto de recursos de redirección que se consideran sinónimos o formas alternativas de referirse a una entidad.
- **Desambiguación**: modela, de la misma forma que sucedía en la Wikipedia, la polisemia, donde cada entidad representa un significado diferente para el mismo término. Se representa a través de la relación `dbo:wikiPageDisambiguates`.
- **Enlaces internos**: el conjunto de enlaces que contiene una página se modela en la DBpedia a través de la relación `dbo:wikiPageWikilink`. Navegando a través de ella podemos encontrar aquellas entidades que mantienen cierta relación con el recurso origen, ya que forman parte de la explicación que aparece en la página de la Wikipedia. Hay que puntualizar que esta es una relación que apunta a gran cantidad de recursos, ya que dentro de un artículo de la Wikipedia hay muchos enlaces a otros artículos que explican los conceptos a los que referencian.

1.3. Análisis de las propuestas

Inicialmente gran parte de las propuestas del estado del arte [12, 68, 25, 71, 40, 34, 32, 37, 88, 39, 66] se han basado en la *Wikipedia* como repositorio de referencia para seleccionar no sólo las entidades a enlazar en el proceso de anotación, sino el conjunto de características léxicas que ayudarían en el proceso de desambiguación. En el momento en que los repositorios semánticos como la *DBpedia* empezaron a dominar la escena, propuestas como [42, 72, 67, 74, 38, 46, 48, 18, 33, 106, 93] han basado su proceso de anotación en este repositorio para hacer uso de las características semánticas que proporcionan las ontologías. Otras soluciones han optado por combinar el conocimiento de la Wikipedia y la DBpedia [28], Wikipedia y WordNet [95, 77, 73], DBpedia y YAGO [104], sólo YAGO [52], o una combinación de Wikipedia, DBpedia y YAGO [44, 80].

Analizamos cada una de las propuestas teniendo en cuenta las tres tareas más importantes que resumen el proceso de EL: la identificación de términos en el texto, la selección de las entidades candidatas del repositorio para realizar la anotación; y la desambiguación de las entidades para identificar aquella que representa al término. Para facilitar la comprensión de cada uno de los enfoques llevados a cabo por cada una, a lo largo de la sección resumimos en tablas un conjunto de características para las propuestas que comentaremos:

(C1) Anotación basada en contexto: identifica el conjunto de términos relevantes que define el contexto del documento y usa estos términos durante el proceso de anotación.

(C2) Desambiguación de entidades: consistiría en la identificación del significado adecuado para cada término según su contexto, en otras palabras, seleccionar la URI de la entidad correcta en la ontología a la hora de realizar la anotación.

(C3) Solución pesada: los conceptos identificados están pesados u ordenados de mayor a menor según su relevancia, permitiendo la selección de las mejores soluciones.

(C4) Exploración del grafo: exploración del grafo del repositorio con el que se realizará la anotación; ya sea exhaustiva, es decir, considerando todas las relaciones del grafo; o limitada en profundidad, desechando aquellos nodos que sobrepasen el límite de exploración establecido.

(C5) Grafo como solución: se considera que la solución del proceso de anotación es un grafo donde se relacionan todas las entidades identificadas.

(C6) Utilización de la DBpedia: uso del repositorio para la obtención de características semánticas en el proceso de enlazado entre las menciones de los términos y las entidades.

1.3.1. Primeras aproximaciones en el estado del arte

En un primer momento, las propuestas que intentaron resolver el problema de NEL se basaron en técnicas de *Procesamiento de Lenguaje Natural* (PLN), donde realizaban extracción de términos clave o un NED muy básico utilizando la Wikipedia [12, 25, 68, 71, 40]. Estas técnicas fueron avanzando hacia el uso de las ontologías donde se modela el conocimiento de forma semántica, entendiéndolas como grafos donde se representan los conceptos como nodos y los arcos como las relaciones entre ellos. En esta línea se produce un cambio de tendencia hacia técnicas de minería de grafos combinadas con técnicas de PLN para realizar tareas de EL, mejorando los resultados.

En el marco de las primeras propuestas listadas en la tabla 1.1, se hace referencia a [12, 25, 68, 71, 40, 52, 42]. Los primeros trabajos que definen el NED como el reconocimiento de entidades y la desambiguación de las mismas usando grandes bases de conocimiento son [12, 25]. Pero en [71] se sientan las bases de la anotación actual refinando esta primera

Propuesta	C1	C2	C3	C4	C5	C6
Bunescu y Pasca [12]	✓	✓	✓			Wikipedia
Mihalcea y Csomai [68]	✓	✓	✓			Wikipedia
Cucerzan [25]	✓	✓				Wikipedia
Milne y Witten [71]	✓	✓	✓			Wikipedia
Han y Zhao [40]	✓	✓				Wikipedia
Kasneci et al. [52]				✓	✓	YAGO
Heim et al. [42]				✓	✓	✓

Tabla 1.1: Comparativa de propuestas de anotación (I).

propuesta, donde no sólo se identifican entidades, sino aquellos términos que son relevantes dentro del texto, convirtiendo texto no estructurado en estructurado debido al enriquecimiento de la información a través de enlaces a conceptos que representan los términos en las grandes bases de conocimiento.

Todas las propuestas abordan el problema de la desambiguación de conceptos utilizando información de contexto para cada uno de los términos, pero entendiéndose por contexto una ventana de términos alrededor del término a anotar. No todas las propuestas calculan una relevancia asociada a los resultados obtenidos. Una aportación interesante fue realizada por [71] con la introducción de la medida de *correlación* entre dos entidades, $r(p_1, p_2)$. Utilizada con frecuencia en otras propuestas posteriores, se define como las relaciones compartidas hacia los mismos conceptos entre dos entidades p_1 y p_2, y se puede entender como la relación o afinidad entre las entidades a través de aquello que comparten entre sí. En [25], se propone representar el conjunto de términos del texto a anotar y las posibles entidades candidatas a desambiguar a través de vectores, utilizando el modelo de espacio vectorial para calcular la relación entre los términos y las entidades usando la medida de similitud del coseno entre los vectores. Mientras que en [40] se propone un sistema de desambiguación utilizando una medida de relación semántica entre entidades (medida de correlación propuesta en [71]) y posteriormente se aplica un algoritmo de clustering jerárquico aglomerativo para seleccionar qué entidad representa el término del texto.

Los principales problemas de estas propuestas son que *(i)* al utilizar la Wikipedia como principal fuente de información para desambiguar, no se aborda el problema usando ontologías, debido a que las grandes bases de conocimiento como DBpedia, Freebase o YAGO estaban en vías de desarrollarse o en una etapa inicial de reciente creación. En este sentido

no se explotan las características semánticas disponibles en las ontologías, por lo que se hace poco uso de la información categorizada y de las jerarquías; o como en el caso de [12], que se limita este uso a un conjunto de categorías de la Wikipedia muy específicas. Otro gran problema al que se enfrentan estas propuestas es *(ii)* la necesidad de conseguir grandes cantidades de datos para entrenar algoritmos de clasificación o clustering, ya que estas propuestas están basadas en la idea de anotación como un problema de clasificación usando algoritmos de aprendizaje supervisado. En la propuesta de [12] se utilizan máquinas de soporte vectorial (SVMs), en [71] se realiza una comparativa de precisión entre diferentes métodos de clasificación como Naïve Bayes, árboles de decisión C4.5 y SVMs; mientras que en [40] se decantan por métodos de clustering de agrupamiento jerárquico. Por último, *(iii)* son aproximaciones conservadoras donde se anota un bajo porcentaje de términos. A modo de ejemplo, en [68] se indica que se anotan un 6 % del total de términos del documento.

Paralelas a estas propuestas están [42, 52], donde se presentan las primeras aproximaciones de exploración de las ontologías como DBpedia, explotando el potencial de los grafos semánticos como solución. En [52] se propone *Ming*, un sistema que descubre nuevo conocimiento extrayendo información de repositorios semánticos dadas dos o más entidades. Se crean subgrafos entre dos entidades mediante heurísticas para explicar la relación existente entre ellas y se mide cómo están de relacionadas utilizando una medida estadística de correlación. Una vez creado el grafo, se utiliza A^* para desechar los nodos que no forman parte de los caminos más cortos y posteriormente se aplica un algoritmo de camino aleatorio para recorrerlo y pesar los nodos con una variación del PageRank [84] para obtener las entidades relevantes. Otra propuesta que trata de explicar la relación entre dos entidades utilizando una solución diferente es la que se describe en [42]. Se presenta *Relfinder* como un sistema de visualización de relaciones entre dos o más entidades. El algoritmo es incluso más sencillo: se buscan relaciones desde la mínima profundidad hasta un máximo establecido y se filtran aquellas no deseadas (configuradas por el usuario).

En estas propuestas se explora el potencial de los grafos y se relacionan dos entidades mediante nodos que proporcionan nueva información; sin embargo no se enfoca el problema como un problema de anotación, cuyo objetivo es la identificación y desambiguación de las entidades. Tampoco se utiliza el texto de entrada, como consecuencia no se genera un contexto para las entidades que se pretenden relacionar, de forma que ayude a afinar los grafos y la recomendación de información; por lo tanto la solución es limitada. En [42] incluso la solución es tan sencilla que ni siquiera se pesan ni se establecen relevancias entre los nodos

Propuesta	C1	C2	C3	C4	C5	C6
Gentile et al. [34]	✓	✓	✓			Wikipedia
Ferragina et al. [32]	✓	✓	✓			Wikipedia
Mirizzi et al. [72]			✓	✓		✓
Hachey et al. [37]	✓	✓	✓	✓		Wikipedia
Ratinov et al. [88]	✓	✓	✓			Wikipedia
Mendes et al. [67]	✓	✓	✓			✓
Hoffart et al. [44]	✓	✓	✓			✓Wikipedia y YAGO
Han et al. [39]	✓	✓	✓			Wikipedia
Muñoz et al.[74]	✓	✓	✓			✓
Meij et al. [66]		✓	✓			Wikipedia

Tabla 1.2: Comparativa de propuestas de anotación (II).

de los caminos que encuentra.

1.3.2. Nuevo enfoque en la resolución de EL

La tabla 1.2 resume un conjunto de propuestas de anotación semántica más recientes que sigue utilizando parte de las técnicas anteriores, pero con ciertas mejoras, como es el caso de [34, 32, 72, 37, 88, 67, 44, 40, 74, 66]. En estas propuestas se empieza a consolidar la medida de similitud del coseno como medida robusta y fiable para calcular la similitud entre un conjunto de términos, y además se continúa utilizando la medida de correlación introducida en [71]. La mayor parte de las propuestas siguen utilizando la Wikipedia como fuente de información, pero algunas comienzan a centrar sus soluciones en la DBpedia, debido a su contenido semántico.

En estas propuestas se introducen los grafos como estructura de datos para representar las relaciones entre los términos identificados en el texto y las posibles entidades a desambiguar. Estos grafos no se utilizan como solución final, sino como un medio en la técnica de desambiguación para obtener una única entidad que represente a cada término. Mediante los grafos se modelan medidas y relaciones ficticias que representan similitud, compatibilidad, relación sintáctica y probabilidades entre el texto y las entidades de los repositorios. Utilizando algoritmos como la detección de caminos aleatorios [96], detección de subgrafos de alta densidad [22] o PageRank para pesar las relaciones o nodos, se obtiene aquella entidad que maximice las medidas y represente unívocamente a cada término, que será la solución de desambiguación. En [34, 37, 88, 44] se trata de resolver el problema de NEL, anotando sólo nombres de

entidades; mientras que en [32, 67, 72, 39, 74, 66] se aborda el problema como EL.

En [34] se utilizan características extraídas de la Wikipedia (frecuencias de apariciones y términos del título del artículo al que hace referencia el concepto) para calcular la relación entre dos entidades. Las entidades que tienen las mismas características se conectan entre sí. Se combina con un modelo basado en caminos aleatorios de un grafo y una matriz de probabilidades para seleccionar el concepto apropiado. Siguiendo la misma idea de probabilidades para representar una medida de selección de entidades, en [32] se presenta *TagMe*, un sistema cuyo objetivo es el enriquecimiento de textos cortos (como los fragmentos de texto proporcionados por los proveedores de noticias) a través de enlaces hacia la Wikipedia. TagMe se basa en seleccionar un conjunto de entidades candidatas obtenidas de un índice creado a partir de los títulos de los artículos de Wikipedia, los textos que enlazan a otros artículos y las redirecciones. Posteriormente asigna una calificación a cada entidad candidata combinando la medida de correlación (obtenida a partir de una representación sencilla en forma de grafo de la Wikipedia) y la probabilidad de enlazar la entidad a un término dentro de los posibles a anotar. Finalmente elimina aquellas entidades que considera que no son coherentes con el resto de entidades del texto. La contrapartida de estas propuestas es que ambas se basan en medidas de correlación y probabilidad sobre la Wikipedia para seleccionar la entidad correcta, dependiendo su solución exclusivamente de este repositorio, sin utilizar la exploración de las relaciones que proporcionan información semántica para desambiguar los candidatos.

En [72] se propone una solución sencilla que, aunque no resuelve el problema de NED, ya que la tarea de desambiguación se realiza de forma manual por el usuario, es una de las primeras en realizar una exploración en el grafo de la DBpedia siguiendo relaciones jerárquicas hasta una profundidad de dos niveles para recomendar términos relacionados al anotado. En [37] se presenta una aproximación para realizar NEL explotando la estructura del grafo que proporcionan los enlaces. En este caso la desambiguación deja de ser manual y pasa a ser automática. Se extrae del texto los posibles términos a anotar utilizando similitud del coseno con los conceptos de la Wikipedia. Se desambiguan creando un subgrafo de profundidad dos usando la Wikipedia, partiendo de los enlaces internos de los artículos y sus categorías para relacionar las entidades candidatas. Para ello se realiza una fase de pesado usando la medida de centralidad de grado, PageRank y la similitud del coseno. En estas propuestas se destacan tres problemas importantes. El primero, la realización de una exploración superficial de la Wikipedia, de un dos niveles en ambas. El segundo problema es la utilización del subgrafo creado sólo como mero instrumento de desambiguación y no como solución para aportar ri-

queza a la anotación semántica. Por último, no construyen un contexto para ayudarse en el proceso de desambiguación: en [37] simplemente se mide la similitud entre el término y el candidato; en [72] es incluso inexistente, ya que esta tarea se realiza de forma manual.

En [88] se trata de hacer una comparativa entre aproximaciones de desambiguación basadas en técnicas locales más tradicionales, [12, 68]; y las basadas en técnicas globales de desambiguación de documentos, [25, 71, 40]. Así, es introducida la desambiguación local como la forma de desambiguar cada mención de forma independiente, utilizando la similitud textual del término y del candidato a desambiguar. Mientras que, la global, trata de desambiguar cada mención de forma simultánea para obtener un conjunto coherente de entidades utilizando las relaciones de los propios conceptos que intenta desambiguar. Como contribución, se propone *Illinois Wikifier*, un anotador basado en desambiguación global, entendida como un problema de optimización que trata de maximizar la coherencia entre entidades. Se extraen los términos del texto utilizando un NER propio [87] y se utiliza una medida de relación entre las páginas de la Wikipedia basada en la medida de distancia de similitud normalizada de Google [20] (*NGD*), que es la que se trata de maximizar. Se concluye que la desambiguación global es prometedora y se puede mejorar, pero la desambiguación local es muy difícil de superar, llegando a la conclusión que la mayor parte de las veces las características globales no son suficientes para predecir si un candidato es el correcto en el proceso de desambiguación. El problema más importante de la técnica de desambiguación global propuesta por [88] es que se basa en la formalización de un problema NP-duro, lo que hace imposible aplicarlo a textos largos con gran cantidad de términos a desambiguar. Tampoco se saca mucho provecho del conjunto de relaciones de la ontología para mejorar los resultados de la medida de coherencia.

En [39] se entiende que los términos de un mismo documento están relacionados y se explota esa interdependencia para realizar NEL de forma colectiva. Se propone la creación de un grafo referencia propio para capturar dos relaciones. La primera, la relación entre el término y la entidad candidata, identificado como compatibilidad y calculada como la similitud entre los contextos del término y la entidad. La segunda, la relación semántica entre las dos entidades, entendida como interdependencia entre entidades, calculada por los enlaces compartidos entre las entidades candidatas de los distintos términos (medida de correlación de [71]). Se utiliza una tercera medida de evidencia, entendida como la importancia de un término dentro del contexto, para propagarla por el grafo y seleccionar la entidad de mayor peso. Mientras que en [44] se sigue una aproximación muy similar con *AIDA*, aunque en este caso se sustituye la medida de evidencia por una reducción del grafo a un subgrafo de alta densidad,

donde cada término está conectado a un único concepto. En [74] se realiza una aproximación de anotación sencilla, calculando un contexto para cada término usando una variación de la medida de [71], midiendo la correlación de los términos en la Web. Utilizando una medida de similitud del coseno entre el contexto creado y el de la entidad, se selecciona el concepto cuya similitud sea mayor. A pesar de ser nuevas propuestas, ninguna enfoca el problema de una forma innovadora, ya que siguen utilizando las mismas aproximaciones que las soluciones propuestas hasta la fecha, con la medida de correlación de [71] como nexo común entre todas ellas.

En cambio, en [67] se presenta *DBpediaSpotlight*, una de las primeras aproximaciones que no utiliza la medida de correlación entre las entidades. Es un anotador que se toma como referencia en el campo, ya que proporciona una infraestructura que combina NEL y EL utilizando enteramente la DBpedia. En una primera fase, se analiza el texto en busca de términos a anotar mediante coincidencia terminológica a partir de un índice creado usando títulos y enlaces de redirección de la DBpedia, de la misma forma que la propuesta anterior. En la segunda fase, se selecciona el conjunto de entidades candidatas para cada término a través de a un mapeo entre los términos obtenidos del índice, con los posibles conceptos de la DBpedia. Para la fase de desambiguación, se utiliza una representación basada en el modelo espacio vectorial de los contextos del término a anotar y del conjunto de entidades candidatas (usando una aproximación de *bolsa de palabras*) y mediante la medida de similitud del coseno se selecciona aquella entidad que tenga mayor similitud.

En [66] se aplica el problema de anotación al enriquecimiento de contenido en entradas de microblogs, aunque no se profundiza en la solución para proporcionar más información que la que ofrece un solo enlace. En esta propuesta se identifican los conceptos que están semánticamente relacionados y se generan enlaces a su correspondiente artículo en la Wikipedia. Se expone que debido a la naturaleza de las entradas cortas de blogs o *tweets*, las aproximaciones propuestas hasta la fecha no son aplicables, por lo que se propone una aproximación basada en aprendizaje supervisado que mediante un conjunto de características obtenidas del contexto del término y del propio usuario, es capaz de decidir si una entidad es relevante. Inicialmente se maximiza la cobertura con un conjunto amplio de candidatos, después se afina la precisión aplicando un clasificador sobre los resultados. Se concluye que la mejor solución es la obtenida por los clasificadores *Random Forest* y *GBRTs* (del inglés *Gradient Boosted Regression Trees*). El problema de esta aproximación es la misma que las propuestas que se han explicado inicialmente: disponer de un conjunto de aprendizaje para entrenar a un clasi-

ficador no es sencillo, ya que este tipo de conjuntos no son habituales. Además es necesario personalizar cada conjunto para el caso concreto de textos con poca información. Por ese motivo, las nuevas aproximaciones han abandonado este enfoque. Tampoco es correcto que se justifique que no se puedan aplicar las técnicas hasta ahora propuestas (PLN, exploración de relaciones semánticas, medidas de correlación entre entidades, similitud de coseno, etc) al enriquecimiento de textos cortos y con poca información, y que por ese motivo se necesiten aplicar algoritmos de aprendizaje; ya que en [32] se realiza la misma propuesta, pero con una aproximación sin aprendizaje, consiguiendo una precisión en sus experimentos del 91,5 % en textos cortos.

En general, una desventaja importante de todas estas propuestas es la utilización exclusivamente de información textual para el proceso de desambiguación de entidades. La selección de las entidades candidatas se realiza siguiendo la misma técnica: selección de un conjunto de entidades que coinciden terminológicamente o con cierta similitud al término que se quiere anotar, añadiendo ciertas medidas adicionales que son comparaciones término a término. Sólo una aproximación mejora la desambiguación mediante el uso de la estructura semántica de la ontología. De todas formas, hay que puntualizar que hasta ahora todas estas aproximaciones anotan un término con una sola entidad, no enriquecen la descripción de cada uno de los términos con un grafo semántico donde los términos se relacionen entre ellos, ya sea para mejorar procesos posteriores como búsquedas o para disponer de más información relacionada, permitiendo extender el contenido inicial.

1.3.3. Minería de grafos y semántica como soluciones al EL

A partir de los trabajos anteriores, el estado del arte cambia y se comienza a utilizar de forma habitual el componente semántico de las ontologías. En este caso, algoritmos como DFS [99], BFS [59] y A^* [41] son los utilizados para explorar el grafo de la DBpedia. El auge de la exploración y creación de grafos crece y se desarrollan algoritmos de anotación que hacen uso de minería de grafos para mejorar la desambiguación en la tarea de NED y EL. En este sentido, medidas como centralidad de grado, cercanía e intermediación ganan popularidad, mientras que algoritmos como HITS [54], PageRank [84], camino aleatorio [61] y sus variaciones son los utilizados para pesar las entidades representadas a través de los nodos. La similitud textual no se abandona y se sigue utilizando la del coseno. Propuestas como [38, 77, 46, 48, 18, 28, 33, 106, 80, 73, 104, 93], listadas en la tabla 1.3 son las más recientes del estado del arte que utilizan información semántica para la desambiguación de

Propuesta	C1	C2	C3	C4	C5	C6
Hakimov et al. [38]		✓	✓	✓		✓
Navigli et al. [77]	✓	✓	✓	✓		Wikipedia y WordNet
Hulpus et al. [46]	✓	✓	✓			✓
Hulpuş et al. [48]	✓	✓	✓	✓	✓	✓
Carvalho et al. [18]	✓	✓	✓			✓
Dojchinovski et al. [28]	(✓)	✓	(✓)			✓y Wikipedia
Fetahu et al. [33]	✓	✓	✓			✓
Varga et al. [106]	✓	✓	✓	✓		✓
Nguyen et al. [80]	✓	✓	✓	✓		✓, Wikipedia y YAGO
Moro et al. [73]	✓	✓	✓	✓		Wikipedia y WordNet
Usbeck et al. [104]		✓	✓	✓		✓y YAGO
Schuhmacher et al. [93]	✓	✓	✓	✓	✓	✓

Tabla 1.3: Comparativa de propuestas de anotación (III).

entidades.

En [38] se propone el anotador *NERSO* para realizar NED basado en la medida de centralidad de grado y una medida de relación semántica entre entidades. Se identifican los términos a anotar en el texto utilizando un algoritmo de n-gramas que, mediante consultas SPARQL a un listado de todas las etiquetas de los recursos, enlaces de redirección y desambiguación de la DBpedia, obtiene una lista de entidades candidatas. Por lo tanto, cada término identificado tiene una lista de recursos candidatos a desambiguar. Posteriormente se construye un grafo con las entidades candidatas y las relaciones inmediatas entre ellas a partir de los enlaces *wikilink* de la DBpedia. Por último, se desambigua calculando para cada nodo su centralidad, utilizando una medida de centralidad modificada que incluye todos los nodos accesibles a través de todos los caminos. Finalmente se devuelve una entidad por cada término detectado cuyo valor de centralidad sea mayor. Hay que puntualizar que para desambiguar una entidad sólo se utiliza la información de la topología del grafo mediante la medida de centralidad, sin tener en cuenta ningún contexto que aporte mayor información al método de desambiguación.

Una propuesta más arriesgada es [77], donde se apuesta por la creación de *BabelNet*, un repositorio semántico que combina Wikipedia y WordNet creado para utilizarse en la resolución de problemas basados en WSD y NED. No se realiza anotación, pero se sigue el mismo proceso de desambiguación para poder enlazar dos entidades idénticas de ambos repositorios. Se utiliza un grafo como resultado de una exploración DFS en un nivel de profundidad 2

siendo los nodos una combinación de ambos repositorios. Enfocando el problema como un problema de ranking, se utiliza la centralidad de grado, otros aspectos de la topología de la red (como longitudes de caminos) y PageRank para pesar los nodos y seleccionar el de mayor peso. El mayor problema de este sistema es la gran cantidad de parámetros configurables a ajustar para obtener una buena desambiguación. Como aplicación práctica de este repositorio, en [73] se propone *Babelfy* para resolver los problemas de NED y NEL. Se extrae de BabelNet el conjunto de entidades a desambiguar para cada término y se enlazan utilizando "firmas semánticas" precalculadas, que no son más que conjuntos de nodos relacionados para una entidad concreta mediante un algoritmo de caminos aleatorios. Se extraen grafos de alta densidad para reducir la ambigüedad aplicando heurísticas. Mediante centralidad y coherencia léxica se pesan los nodos para relacionar los términos con el de mayor valor.

En propuestas como [46, 18, 28, 33] se trata de resolver otros problemas relacionados al de NED, como WSD, detección de hiperónimos, representación de relaciones en un texto o clasificación. En [46] se propone una técnica para WSD utilizando la centralidad del vector propio (variación del algoritmo HITS) sobre un grafo para calcular la relación entre las combinaciones de los posibles candidatos a desambiguar mediante una matriz de adyacencia. En [28] se introduce el problema de detección de hiperónimos como un sistema de clasificación no supervisado de entidades donde se identifican los términos más relevantes del texto y se enlazan con una lista de términos y conceptos de la DBpedia más genéricos con diferentes niveles de granularidad. [18] y [33] son meta algoritmos, donde se trata de reutilizar la salida del proceso de NEL de *DBpediaSpotlight* para representar las relaciones de un texto (ya sean nombres, acciones o lugares). En el caso de [18], estas relaciones se representan a través de un grafo. En cambio en [33], se utilizan para clasificar un texto, creando metadatos semánticos en conjuntos de datos para mejorar las búsquedas en repositorios. Estas propuestas no aportan un sistema innovador, sino que apuestan por reutilizar un sistema de anotación existente, con pocos añadidos, para proporcionar una solución rápida a los problemas que intentan abordar, pero con una salvaguarda: dar valor al proceso de anotación.

En [48] se presenta *Kanopy*, un sistema de anotación semántica y clasificación, donde se realiza NED, NEL y extracción de subgrafos para etiquetar el texto mediante categorías. Es la primera aproximación que da valor a la creación de un subgrafo extraído de la DBpedia. Es una solución muy completa donde a partir de los términos detectados en el texto, se agrupan mediante un algoritmo de agrupamiento aglomerativo y se desambiguan. A partir de estas entidades desambiguadas, se explora en profundidad de nivel dos el grafo de la DBpedia para

categorizar los conceptos iniciales y establecer relaciones entre las categorías. La exploración y el enriquecimiento mediante la DBpedia se explica en [47]. Cada entidad desambiguada tiene un subgrafo de categorías creado mediante una exploración de nivel dos, donde una vez construido se pesan los nodos aplicando medidas de centralidad para determinar la importancia que tienen respecto al término que intentan anotar y al clúster de términos relacionados creado en la etapa inicial. Tendrá más peso aquel nodo que sea más importante y central en relación al grafo, por lo tanto será esa categoría la que represente al conjunto de términos. En resumen, el objetivo final de Kanopy es utilizar el grafo para proporcionar categorías relacionadas y permitir al usuario ver cómo están relacionados los términos a través de ellas. En este caso concreto no se intenta representar las entidades importantes del texto en forma de grafo y extender su contenido con información relacionada, sino que se trata de resolver un problema de categorización y etiquetado ayudado de representaciones en forma de grafo. También se ve acotado en la exploración del mismo, porque se limitan las relaciones a explorar a relaciones puramente jerárquicas de la ontología en una profundidad máxima de dos niveles.

En la misma línea de categorización de texto tenemos la propuesta de [106], en la que se aplica la exploración de grafos a un problema de clasificación de micropost (*tweets*) según temática para clasificar automáticamente mensajes cortos en dos clases: "respuesta de emergencia" y "detección de violencia". Se introduce una estructura de grafo pesada alrededor de los conceptos del mensaje para enriquecer el contexto usando los repositorios disponibles. Como repositorios de referencia se utiliza el propio Twitter, la DBpedia y Freebase. Se analizan utilizando algoritmos de bolsas de palabras, capturando la importancia de cada término usando TF-IDF. La fase de desambiguación se lleva a cabo haciendo EL a través de anotadores externos existentes para enlazar las entidades con los repositorios: OpenCalais [89] y Zemanta [101]. Se utiliza la información semántica de los repositorios DBpedia y Freebase para crear meta grafos semánticos de profundidad uno (relaciones inmediatas) donde se extraen una serie de características para aprovechar la información semántica: clase de las entidades y categorías semánticas de la DBpedia. Se pesa cada una de estas características según su especificidad/generalidad para una entidad o en relación al repositorio y se utilizan estos valores para entrenar un clasificador SVM, que es el que proporcionará la temática asociada al texto de partida una vez haya sido entrenado.

En [80] se presenta *AIDA-light*, nueva versión de *AIDA* [44] alcanzando una precisión más alta y mejorando tiempo de respuesta y escalabilidad. En AIDA-light se usa una variación del algoritmo en dos etapas: una primera de resolución fácil donde se identifican entidades de

baja ambigüedad y se enlazan a los recursos, aprovechando para definir el dominio del texto mediante un diccionario entidad-dominio creado apriori de forma manual. De esta forma, se establece un contexto de desambiguación para el resto de entidades difíciles, pertenecientes a la segunda etapa de desambiguación, que sirve para acotar los posibles resultados en el resto de entidades. En esta segunda fase, se implementa una medida de coherencia entre entidades y coherencia entre entidad y dominio para pesar las entidades candidatas y seleccionar la correcta. Teniendo en cuenta lo anterior, se crea un grafo donde los nodos son los términos y las entidades candidatas, mientras que las relaciones entre ellos están pesadas a través de las medidas de coherencia comentadas. El objetivo es encontrar el mejor subgrafo que mapee cada término con una sola entidad, que será devuelta por el algoritmo.

Una propuesta que explora hasta un nivel tres de profundidad utilizando BFS es *AGDISTIS* [104], una aproximación de NEL, basada en medidas de similitud, expansión heurística de los términos a anotar y un algoritmo basado en grafo donde se aplica HITS para pesar los nodos y desambiguar las entidades. La principal desventaja de esta aproximación es la falta de un sistema de detección de términos a anotar, ya que se ha enfocado como una propuesta puramente de desambiguación de entidades, donde el sistema parte de un conjunto de términos iniciales que son los que se desambiguarán.

Otra propuesta interesante es [93] donde se utiliza un modelo semántico basado en grafo para representar el contenido de un texto, desambiguando entidades e introduciendo relaciones semánticas entre ellas. Se explotan los beneficios de la representación semántica para calcular la similitud entre documentos, comparando los grafos semánticos, simplificando la tarea de NED al usar *DBpediaSpotlight* para obtener las entidades iniciales para explorar y crear el grafo. Al igual que propuestas anteriores, se crea el grafo a partir del algoritmo DFS a una profundidad de dos niveles. Se propone una medida para pesar las relaciones y cuantificar el grado de relevancia de éstas con respecto a las entidades que conectan. Es una medida de probabilidad que asigna más peso a los arcos más específicos. Para pesar los nodos, se calcula el coste del camino mínimo desde el nodo inicial (la entidad desambiguada) hasta el resto de nodos para establecer un ranking de pares de nodos según esa medida de distancia semántica. Como aportación principal, se formula una medida basada en distancias de grafos para calcular la similitud entre documentos. Hay que recalcar que la principal utilidad del grafo es para comparar documentos, no tanto para enriquecer el contenido ni recomendar información. También hay que tener en cuenta que al pesar las relaciones y los nodos mediante el coste del camino mínimo no se tiene en cuenta un contexto, sino que siempre se toma como referencia

el nodo inicial, la entidad desambiguada, y se pesa en relación a ese nodo, pudiendo generar un contenido que se aleja de la idea general del texto y solamente se relaciona con un solo término.

Destacan dos problemas importantes en todas estas propuestas. El primero, la realización de una exploración superficial del grafo de la DBpedia (o Wikipedia), hasta una profundidad máxima de dos niveles en [77, 48, 93] y tres en [104], siendo el nivel uno el utilizado por el resto de las propuestas. El segundo problema es la utilización del grafo sólo como mero instrumento de desambiguación, no como solución para aportar enriquecimiento de información más allá de la anotación semántica. Solamente propuestas como [93], que lo han utilizado como forma para medir la similitud entre documentos, o [48] como forma de aportar una jerarquía de clasificación, han dado una utilidad al grafo más allá del establecido para desambiguar una entidad.

1.4. Conclusiones

A pesar del esfuerzo investigador durante esta última década, las aproximaciones de anotación semántica se siguen enfrentando a importantes desafíos que afectan en gran medida a la calidad de sus resultados. El objetivo es, a simple vista, sencillo y consiste en identificar los términos relevantes de un documento y asociarlos con su representación semántica dentro de una ontología. Sin embargo, como se ha detallado a lo largo de este capítulo, las aproximaciones analizadas siguen teniendo importantes dificultades para establecer correctamente esta asociación. Este problema se puede apreciar en las distintas problemáticas que intenta resolver la anotación semántica, es decir, (*i*) el reconocimiento, (*ii*) la desambiguación y (*iii*) el enlazado de entidades. Por un lado, las principales soluciones al problema de reconocimiento de entidades hacen uso de técnicas de PLN y por ello comparten sus ventajas e inconvenientes. Son, por lo tanto, aproximaciones muy eficientes y generalmente basadas en diccionarios de datos, pero que no consiguen aprovechar la información semántica disponible. Precisamente, las técnicas de desambiguación y enlazado de entidades tratan de paliar este problema buscando la entidad de la ontología que mejor se empareja con el término del documento.

Sin embargo, las aproximaciones existentes en la literatura no consideran, o solo lo hacen parcialmente, un aspecto clave a la hora de establecer el correcto significado de un término: el contexto en el que se utiliza dicho término. En este paradigma, el contexto de un documento representa al conjunto de términos clave que permiten identificar la temática del documento

analizado. Proporciona, por lo tanto, información imprescindible para la correcta desambiguación de aquellos términos en los que puede haber cierta ambigüedad, como por ejemplo entidades con la misma representación sintáctica. No obstante, como se ha detallado, en el mejor de los casos, el uso de ese contexto se limita a un subconjunto de términos relevantes y en ningún caso utiliza la información semántica proporcionada por la ontología. En este sentido, al igual que el texto proporciona un contexto de términos, la ontología, a través de las entidades asociadas a dichos términos, proporciona un contexto que toma la forma de un grafo que enlaza a dichas entidades a través de relaciones. Esta información es clave para mejorar los problemas de anotación semántica mencionados anteriormente, aunque conlleva nuevos problemas a solucionar, como el análisis masivo de información o la posibilidad de ruido, que pueden empeorar los resultados obtenidos. Precisamente, en los siguientes capítulos se profundiza en esta idea y en su integración dentro del marco de anotación y enriquecimiento semántico desarrollado en esta tesis.

Capítulo 2

ADEGA: Algoritmo de anotación semántica para el enriquecimiento de documentos

2.1. Introducción

Como hemos adelantado en el estado del arte, la mayor parte de las aproximaciones usan datos enlazados para anotar términos con un sola entidad de la ontología. Estas aproximaciones hacen uso de información textual para desambiguar qué entidad representará la anotación de un término. Algunas aproximaciones recientes [48, 93, 104] exploran relaciones hacia otras entidades para mejorar la precisión de anotación, extrayendo subgrafos de los datos enlazados. Estas propuestas mantienen la hipótesis de que un grafo de entidades proporciona mucha más riqueza semántica que sólo una entidad.

La propuesta que presentamos en este capítulo sigue el mismo principio: anotar los términos de un documento con grafos extraídos de una ontología. Al igual que los términos de un párrafo o de un documento definen el contexto en el que se está usando un determinado término, nuestra hipótesis es que una entidad contextualizada en una ontología puede enriquecer la calidad semántica de la anotación. Sin embargo, nuestra propuesta de anotación difiere en cómo se obtienen los grafos.

En nuestra aproximación comenzamos identificando aquellas entidades que representan a los términos relevantes del documento, que serán consideradas nodos raíces del grafo. Apli-

camos sobre estos nodos una búsqueda en profundidad a través del grafo de la ontología filtrando aquellas entidades que no están relacionadas con los términos que queremos anotar. La estrategia de exploración del grafo se extiende a las relaciones taxonómicas de la ontología, teniendo en cuenta muchas más entidades a filtrar: nodos hijos, padres o abuelos del nodo a explorar. Finalmente aportamos un método de evaluación de nodos, que considera que no todas las relaciones aportan la misma información, por lo tanto no pueden considerarse que tengan el mismo peso, tienen que ser pesadas en consecuencia para determinar la relevancia de un nodo, al contrario que otras soluciones basadas en grafos.

En resumen, proponemos *ADEGA*, una infraestructura para anotar documentos con grafos que son contextualizados dentro de la temática del documento que anotan, complementando el contenido del documento con información adicional.

2.1.1. *ADEGA*: infraestructura de anotación semántica

El objetivo de *ADEGA* como infraestructura de anotación es proporcionar información complementaria a los usuarios a través de las anotaciones, no solo datos que son procesados por una máquina. En este caso, ontologías como DBpedia ayudan a conseguir este proceso de enriquecimiento. De hecho, las relaciones de una entidad describen propiedades específicas con una semántica, cuyos nodos destino pueden ser o bien datos como cadenas de caracteres, u otras entidades que también están descritas por otras propiedades. Estas conexiones entre entidades configuran una estructura basada en grafo donde el usuario puede navegar fácilmente para obtener más información sobre un concepto específico. La figura 2.1 proporciona una representación visual de este proceso de enriquecimiento. En el dibujo, los términos relevantes del documento se anotan con grafos extraídos de una ontología de datos enlazados, permitiendo navegar por las relaciones y visualizando (por ejemplo, por medio de una plantilla web) la información contenida en las diferentes entidades del grafo.

El proceso de anotación de documentos consiste en adjuntar ciertos metadatos como frases, comentarios o etiquetas a un documento o a parte de él [51]. La anotación semántica extiende este concepto y va un paso más allá para reducir el espacio entre el lenguaje natural y su representación computacional, tratando de emparejar los términos de un documento con su representación semántica; es decir, asociar términos a una entidad de una ontología, encargada de representar el conocimiento de forma estructurada [53]. Aun así, el mismo término puede tener múltiples significados o incluso términos distintos pueden hacer referencia al mismo concepto. Pongamos un ejemplo con el término *Paris* en un documento, donde puede

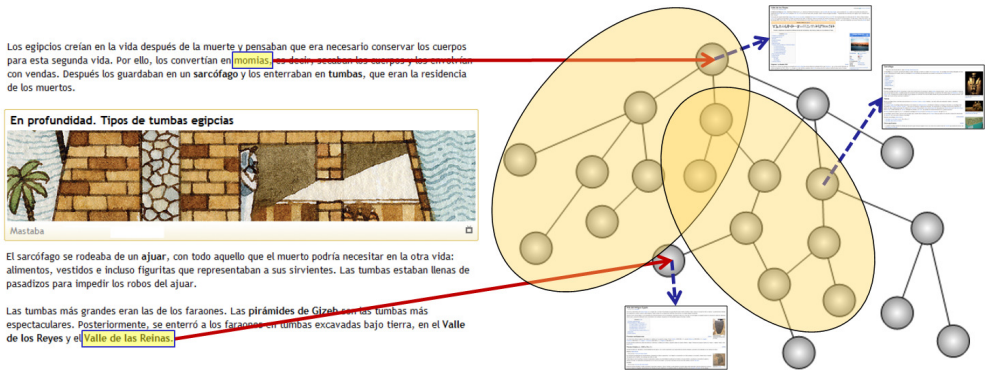

Figura 2.1: Enriquecimiento semántico de los términos relevantes de un documento.

representar la capital de Francia, un personaje mitológico griego o un nombre de persona en la cultura anglosajona, entre otras cosas. Encontrar en este caso la entidad correcta en una ontología es esencial para proporcionar una anotación correcta. Para conseguir este propósito, los sistemas de anotación explotan el *contexto* de un documento, es decir, sus términos relevantes para aumentar la precisión durante este proceso de búsqueda. Por ejemplo, si el término *Francia* o *capital de Francia* apareciese en el texto, la identificación de la entidad correcta sería mucho más sencilla y descartaría otras opciones. Siguiendo el mismo ejemplo, en el caso de *Paris*, existiría una relación que indicase que es la capital de *Francia*.

El principal objetivo de la infraestructura desarrollada, representada en la figura 2.2, es realizar un proceso de anotación semántica del documento de entrada que nos permita seleccionar el grafo que enriquece y describe semánticamente el contenido del documento. Podemos distinguir tres grandes etapas para llevar a cabo este objetivo:

- **Extracción de contexto**. En esta etapa, el documento es analizado combinando técnicas de procesamiento de lenguaje natural con medidas de similitud semántica [19] para obtener una lista de términos relevantes, llamada *contexto*, que caracteriza el contenido del texto. Los documentos de entrada pueden ser desestructurados o tener una estructura concreta. En este último caso, no es lo mismo localizar un término en el título de un documento que dentro de un párrafo de una subsección. Por lo tanto, considerar dónde se ubica un término dentro del documento puede darnos información sobre su relevancia. A esta localización la denominaremos *campo* del término. Finalmente, se ordena cada uno de los términos identificados de acuerdo a su frecuencia y al campo en donde

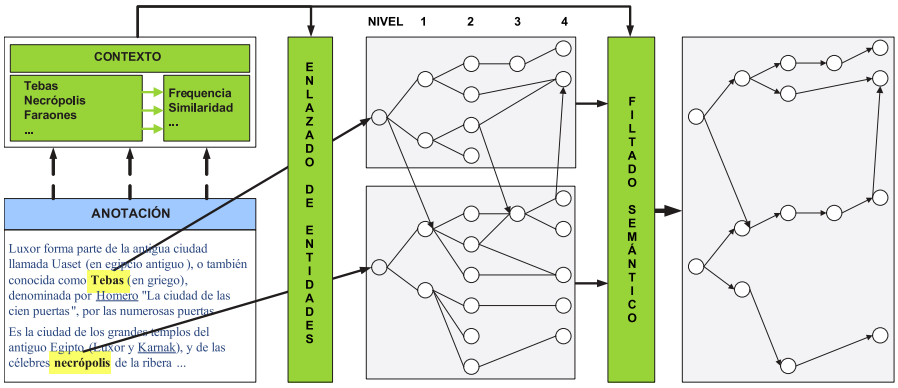

Figura 2.2: Infraestructura de anotación semántica.

esté localizado.

- **Enlazado de entidades**. Para cada término del contexto se invoca una búsqueda, un servicio externo de consultas en un punto de acceso SPARQL en la ontología de la DBpedia, para obtener así una lista de entidades (o URIs) candidatas para anotar a un determinado término. Para seleccionar la entidad más adecuada de entre la lista de posibles candidatas, se aplica un algoritmo de desambiguación de entidades.

- **Filtrado del grafo basado en contexto**. Las entidades identificadas en el paso anterior se utilizan como nodos raíz del proceso de exploración del grafo. El grafo se obtiene explorando exhaustivamente las relaciones y entidades de la ontología y basándose en la ocurrencia de los términos del contexto dentro de las relaciones que parten de las entidades, se selecciona aquellas que son relevantes. Como resultado, se obtiene un (sub)grafo de datos enlazados relacionados con el nodo raíz. Finalmente el grafo se almacena en un repositorio donde el nodo raíz es enlazado al término del documento anotado.

Como resultado de este proceso, el documento de anotación se almacena. En la infraestructura consideramos que la anotación de un documento está compuesta por la anotación de sus términos relevantes; aunque en la práctica, se puede asumir que el documento es enriquecido por un conjunto de grafos, ya que desde cada término anotado cuelga un subgrafo creado específicamente para el contexto del documento.

2.2. Extracción del contexto

La primera tarea de *ADEGA* es la extracción de términos relevantes de un documento, es decir, del *contexto*. En esta tesis asumimos que estos términos capturan la temática del contenido del texto, por lo que pueden ayudar a descartar información de la DBpedia que no está relacionada con el documento. La extracción de contexto se lleva a cabo mediante la ejecución secuencial de tres fases: análisis de morfología, de similitud y de frecuencia. Como resultado, se obtiene un conjunto de términos que pertenecen al contexto, ordenados de acuerdo a su relevancia dentro del texto.

2.2.1. Análisis de morfología

El análisis de morfología determina la categoría gramatical de cada palabra. Para llevar a cabo el análisis, la solución propuesta se basa en la herramienta GATE[1] [26]. La detección de morfología afecta a la creación del contexto realizando las siguientes tareas:

- **Eliminación de términos que no son representativos para caracterizar el contenido**. No se consideran palabras cuya morfología sea la de un verbo, conjunción, preposición o determinante.

- **Detección de términos compuestos**. Términos como *Edad de Piedra* u *Homo Sapiens* no se pueden considerar por separado y se detectan mediante expresiones regulares a través del sistema de definición de reglas JAPE que proporciona la herramienta. Por ejemplo, la creación de la regla *Entity* representada en la figura 2.3 detecta estas expresiones a través de dos macros personalizadas: por un lado la macro `NE_NOUN` donde detectamos términos compuestos como *Venus de Willendorf* y por otro, la macro `ROMAN_NOUN` encargada de reconocer nombres con números romanos, como *Luis XVI*.

2.2.2. Análisis de similitud

Dado que un término puede aparecer de formas diferentes, se crean conjuntos de similitud terminológica con la finalidad de incrementar la ocurrencia de una palabra y así evitar

[1] GATE es un software libre de código abierto para el procesamiento de lenguaje natural que ha sido usado, y se sigue utilizando, en muchos proyectos internacionales. Para más información, consultar https://gate.ac.uk/projects.html

```
Macro: NE_NOUN
(
(UPPERCASE_NOUN|UPPERCASE_ADJ)(UPPERCASE_NOUN)+ |
(UPPERCASE_NOUN) (Token.string==~"[Dd]e"|
Token.string==~"[Dd]el")
(UPPERCASE_NOUN|UPPERCASE_ADJ)|
(UPPERCASE_NOUN) Token.string==~"[Dd]e"
Token.string==~"[Ll][oa]s"
(UPPERCASE_NOUN|UPPERCASE_ADJ)|
Token.category==ADJ (UPPERCASE_NOUN)
)
Macro: ROMAN_NOUN
(
Token.length >1 (Token.orth == allCaps |
Token.length == 1, Token.orth == upperInitial)
)
Rule: Entity
(
(COMPOSITE_NOUN) | (ROMAN_NOUN)
):entity
-->
:entity.Entity = rule = ''Entity''
```

Figura 2.3: Reglas JAPE.

variaciones del mismo término que comparten la misma raíz. En esta propuesta se utiliza SoftTFIDF [91] para calcular la similitud entre palabras, combinando la medida TF-IDF con la función de distancia Jaro-Winkler [109]. El uso de SoftTFIDF se justifica por ser la métrica que mejor resultado obtiene al comparar términos con una raíz común [21].

Para cada documento se entrena SoftTFIDF usando el corpus de términos extraídos durante el análisis morfológico, adaptando así la medida a los términos en los que se va a aplicar. Para seleccionar el término que representa a cada conjunto se siguen los siguientes criterios:

1. Si hay un único *nombre propio* dentro del conjunto, el término es considerado como término nominal. Por ejemplo, *Egipto* es el término nominal del conjunto {*Egipto*, *egipcio*, *egiptología*}.

2. Si hay más de un nombre propio, el término con la frecuencia más alta es considera-

do como el término nominal del conjunto. La frecuencia de estos términos se calcula usando la ecuación 2.1, que será detalla en la subsección 2.2.3.

3. Si no hay un nombre propio, se selecciona el término con la frecuencia más alta dentro del conjunto. Esta frecuencia también se calcula siguiendo la ecuación 2.1.

2.2.3. Análisis de frecuencia

El análisis de frecuencia proporciona la frecuencia relativa de un término t en un *campo* del documento estructurado, α. Para calcularlo también se considera el conjunto de similitud de un término t en α:

$$tf_{t,\alpha} = \frac{num_{t,\alpha} + sim_{t,\alpha}}{|\alpha|} \tag{2.1}$$

donde:

- $|\alpha|$ es el número de términos en α.
- $num_{t,\alpha}$ es el número de ocurrencias de t en α.
- $sim_{t,\alpha}$ es el número de ocurrencias de los términos que son sintácticamente similares a t; y que se han obtenido tal y como se describe en la sección 2.2.2.

2.2.4. Identificando el contexto

Se ordenan los términos en función de su relevancia en el documento usando la medida TF-IDF [63], pero adaptada a las características del propio documento. La relevancia r de un término t en el documento se calcula como:

$$r_t = \left(\sum_{\alpha \in d} tf_{t,\alpha} \cdot p_\alpha \right) \cdot idf_t \tag{2.2}$$

donde:

- d es un documento.
- α es un campo, una localización dentro del documento d.
- $p_\alpha \in [0,1]$ es el peso correspondiente asociado al campo α en el documento.

Término	Relevancia
Paleolítico	0.9328
Homo Erectus	0.8566
Prehistoria	0.6415
Homo Sapiens	0.6173
Neandertal	0.5997
Humano	0.5898
Australopithecus	0.5254
Fósil	0.4888
Edad del Hierro	0.4724
Homo Habilis	0.4410

Tabla 2.1: Contexto de anotación para un documento con temática *Paleolítico*.

- idf_t es la frecuencia inversa del documento y mide la relevancia de un término de un documento dentro de la colección. Los términos raros tienen una contribución mayor en la frecuencia inversa. Como colección consideramos la Wikipedia, porque es un recurso valioso que se utiliza como corpus de documentos en una gran variedad de tareas en el ámbito de la recuperación de información. Además, su elección también ha sido influenciada por la estrecha relación que tiene con la DBpedia. La frecuencia inversa del documento se calcula como:

$$idf_t = log \frac{|D|}{1 + |\{d \in D, tf_t \neq 0\}|} \tag{2.3}$$

  siendo:

  - $|D|$ el número de documentos del corpus de la Wikipedia.
  - $|\{d \in D, tf_t \neq 0\}|$ es el número de documentos donde aparece el término t.

Con esta medida de relevancia, un término del documento alcanza una relevancia alta con una frecuencia terminológica alta y una frecuencia de documento baja en el corpus.

Finalmente se evalúan y ordenan los términos del documento siguiendo la ecuación 2.2. Los términos con $r_t > \theta$ se consideran relevantes y son incluidos en el contexto, siendo θ el umbral de relevancia. La tabla 2.1 muestra la relevancia de algunos términos incluídos en un contexto de un documento cuya temática es el *Paleolítico*.

2.3. Filtrado del grafo basado en contexto

El filtrado es una de las tareas más importantes cuando se accede a grandes repositorios de información, ya que los usuarios quieren ver lo relevante sin tener que bucear entre un mar de datos para encontrarlo. Teniendo esto en cuenta, el primer paso al enfrentarse a grandes cantidades de información, como los datos enlazados, es separar el trigo de la cizaña. En el ámbito de esta tesis, el trigo son las entidades enlazadas con alguno de los términos del contexto, aquellos que caracterizan el documento; mientras que la cizaña es todo lo demás. Pero ¿cómo decidimos si una entidad está verdaderamente relacionada con el contenido del texto?

En esta tesis asumimos que la relevancia de un nodo en un grafo RDF viene dada por sus relaciones. Así, distinguimos distintos tipos de relaciones en la DBpedia: por un lado existen relaciones cuyo rango es de tipo `rdfs:Literal`, es decir, *datos primitivos*. Por ejemplo, todas las entidades de la DBpedia tiene siempre una etiqueta textual representada a través de la propiedad `rdfs:label`. Por otro lado, existen relaciones hacia otros *recursos*, que en el caso de la DBpedia son otras entidades. Por ejemplo, la relación `dbp:familia` entre una entidad que representa al concepto *Homo Sapiens* (identidicada por la URI `dbo:Homo_sapiens`) y su subfamilia de simios, los *homínidos* (identificado por la URI `dbo:Hominidae`).

Independientemente de la aproximación seguida para evaluar los nodos del grafo (detallada en las siguientes subsecciones) consideramos que el peso o influencia de las relaciones no es uniforme. Por ejemplo, la propiedad `dbo:abstract` de una entidad tiene como rango un texto cuyo contenido coincide con la descripción del concepto que aparece en la página HTML del artículo de la Wikipedia. Esta propiedad proporciona mucha más información que otras para discernir si el concepto es relevante, como por ejemplo `dbprop:caption`, que contiene la leyenda de una imagen asociada al concepto que representa.

Teniendo esto en cuenta, se pesan las relaciones establecidas entre dos entidades o entre una entidad y datos primitivos. El valor de los pesos oscila entre 0,0 y 1,0, donde 1,0 representa el valor máximo de importancia. Hay que recalcar que dada la gran cantidad de relaciones definidas en la DBpedia, o en la mayoría de las ontologías, se hace prácticamente inviable para un sistema definir un peso específico para cada una de ellas desde el comienzo. El principal problema en este punto es tener que establecer los pesos de antemano, ya que son necesarios para poder evaluar la relevancia de un nodo. Incluso en un proceso de aprendizaje o ajuste de los pesos, se requeriría una gran cantidad de datos para el entrenamiento.

Relación	Peso pre-establecido
rdfs:label	0.70
dbpedia-owl:abstract	1.00
dcterms:subject	0.90
is dcterms:subject of	1.00
rdf:type	0.60
is rdf:type of	1.00
skos:broader	0.60
is skos:broader of	1.00
rdfs:subClassOf	0.60
is rdfs:subClassOf of	1.00
default	1.00

Tabla 2.2: Mejor configuración de pesos para las relaciones de la DBpedia.

Por esta razón, hemos seguido una aproximación incremental más adecuada donde una nueva relación es añadida en cada iteración; por lo tanto, el problema ahora es encontrar la mejor configuración de pesos para las relaciones seleccionadas. Hay que tener en cuenta que este proceso depende (*i*) de los expertos encargados de establecer los pesos o (*ii*) si este proceso es automático, del número de ejemplos disponibles para entrenar el algoritmo de optimización. Por ejemplo, en los experimentos detallados en la sección 2.4, usamos 10 propiedades de la DBpedia, aquellas que en nuestra experimentación preliminar fueron las más relevantes desde la perspectiva de nuestra aplicación.

Cada relación ha sido pesada siguiendo la aproximación descrita en [107], pero para las relaciones seleccionadas en nuestro sistema. En esa propuesta se usa un algoritmo genético (un método de optimización) para determinar el peso más adecuado para el conjunto de relaciones. Los resultados muestran una mejora en las curvas de precisión y cobertura obtenidas para el conjunto de datos del experimento. Los pesos de la mejor configuración obtenida a partir de estos experimentos están listados en la tabla 2.2. En la sección 2.4.2 se detallan los efectos de diferentes pesos en los resultados de precisión y cobertura del sistema.

2.3.1. Algoritmo de filtrado

La tercera etapa en nuestra solución conceptual (ver sección 2.1) es la aplicación del algoritmo DFS [99] (del inglés *Depth-First Search*) sobre cada entidad recuperada en el módulo de identificación y enlazado de entidades. El algoritmo 2.1, recorre los nodos del grafo de

forma que va expandiendo los nodos hijos hasta que no hay más nodos que expandir o se alcanza el límite de profundidad definido, λ_d. En ese caso, regresa al nodo más reciente que no ha explorado.

El algoritmo devuelve el grafo λ que contiene los nodos relevantes para el contexto Δ empezando desde el nodo n_s. En la primera iteración, n_s es el nodo devuelto en la etapa de *enlazado de entidades* y ϕ_n, ϕ_c, ϕ_t son listas vacías. Estas tres listas se usan para reducir la exploración de los nodos que ya han sido procesados. Específicamente, ϕ_n se usa para almacenar los nodos visitados, mientras que ϕ_c y ϕ_t para las categorías y tipos respectivamente.

El algoritmo tiene dos partes: en la primera, se exploran o visitan las relaciones r de los nodos procesados n_s; mientras que en la segunda, se evalúa n_s. De este modo, en la exploración se siguen las siguientes reglas:

- Sólo se procesan relaciones que tengan una cardinalidad dentro de un umbral establecido, δ_i, esto es, relaciones donde $|r(n_s,\_)| < \delta_i$. Si el número de instancias de la relación entre n_s y r supera este umbral, consideramos que esta relación es demasiado genérica para proporcionar información relevante sobre el nodo.

- Cada instancia de la relación r se procesa recursivamente (línea 5). El grafo resultante $\text{GFA}(n_t)$ se añade a la solución λ. Nótese que $\text{GFA}(n_t)$ puede ser un grafo vacío cuando n_t no tiene relevancia para Δ.

- Si la relación enlaza el nodo con una entidad de categoría o tipo (lineas 6-22 y 23-32, respectivamente), también exploramos las entidades hermanas, padres e hijas de la entidad. El umbral δ_i también se aplica a los elementos que exploramos, ya que se considera que mucha información penaliza en la exploración tanto en calidad de resultados como en tiempo de computación.

2.3.2. Evaluación de nodos

La relevancia de un nodo n se establece en la ejecución de $rel(n,\Delta)$ en el algoritmo 2.1 (línea 36). Como hemos mencionado previamente, el cálculo de esta relevancia depende del tipo de nodo, es decir, si es un datos primitivo o una entidad. En el caso de *entidades*, la relevancia es calculada añadiendo la relevancia de cada una de sus relaciones:

$$rel(n,\Delta) = \frac{1}{|R_n|}\sum_{i=1}^{|R_n|}\left(p_i \frac{1}{|S_{i,n}|}\sum_{t=1}^{|S_{i,n}|} rel(n_t,\Delta)\right) \tag{2.4}$$

Algoritmo 2.1: Graph Filtering Algorithm (GFA)

Input: Node n_s, visited nodes ϕ_n, visited categories ϕ_c, visited types ϕ_t, search context Δ.
Output: The graph solution λ which root node is n_s.
Data: Depth limit δ_d, Instance exploration limit δ_i, Relevance threshold δ_r.

```
1  add node n_s to the visited node list φ_n;
2  for each relation r such that r(n_s, n_t) do
3      if |r(n_s, _)| < δ_i then
4          for each node n_t such that r(n_s, n_t) and n_t ∉ φ_n do
5              add GFA(n_t) to λ;
6              if r = dcterm:subject then
7                  add node n_t to λ;
8                  add category n_t to φ_c;
9                  if |dcterm:subject(_, n_t)| < δ_i then
10                     for each node n_i such that dcterm:subject(n_i, n_t) and n_i ∉ φ_n do
11                         add GFA(n_i) to λ;
12                     end
13                 end
14                 for each category c such that skos:broader(c, n_t) or skos:broader of(n_t, c) and
                   c ∉ φ_c do
15                     add category c to φ_c;
16                     if |dcterm:subject(_, c)| < δ_i then
17                         for each node n_i such that dcterm:subject(n_i, c) and n_i ∉ φ_n do
18                             add GFA(n_i) to λ;
19                         end
20                     end
21                 end
22             end
23             if r = rdf:type then
24                 for each class t such that rdfs:subClassOf(t, n_t) or rdfs:subClassOf(n_t, t) and
                   t ∉ φ_t do
25                     add category t to φ_t;
26                     if |rdf:type(_, t)| < δ_i then
27                         for each node n_i such that rdf:type(n_i, t) and n_i ∉ φ_n do
28                             add GFA(n_i) to λ;
29                         end
30                     end
31                 end
32             end
33         end
34     end
35 end
36 if rel(n_s, Δ) ≥ δ_r then
37     add n_s to λ;
38 end
```

donde:

- Δ es el contexto del documento.
- $|R_n|$ es el número de las diferentes relaciones en el dominio de n.
- i identifica la i-ésima relación de n.
- p_i es el peso de i, $0 \leqslant p_i \leqslant 1$.
- $S_{i,n}$ es el número de instancias de i en el dominio de n.
- t identifica a la instancia t-ésima de i, esto es, $r(n, n_t)$.
- n_t es el nodo objetivo de la t-ésima instancia de i.

La fórmula tiene dos sumatorios. El sumatorio exterior itera sobre cada relación de n y multiplica su peso p_i por su relevancia. La relevancia de una relación i se calcula como la media aritmética de la relevancia de los nodos n_t para los cuales existe una instancia de la relación entre n y n_t. El punto subyacente de la fórmula es que cada relación se evalúa sólo una vez; la razón es limitar la influencia de cada una. Por ejemplo,supongamos un nodo con nueve relaciones diferentes con un único valor y la décima con 10 valores diferentes (relación multivaluada). Si la relevancia del nodo objetivo de cada instancia de la relación es añadida independientemente de si la relación es o no multivaluada, esta décima relación puede sesgar el resultado final. Concretamente, si la décima relación es "nombre del personaje" y varios de los nombres incluyen un mismo término del contexto, probablemente el nodo evaluado será incluido en el grafo final, independientemente del resto de relaciones que pueda tener. Por esta razón, cada relación tiene que contribuir lo mismo.

Para los *datos primitivos* textuales, la relevancia viene dada por los términos compartidos con el contexto:

$$rel(n,\Delta) = \frac{shared(n,\Delta)}{|\Delta|^2} \sum_{i=1}^{|\Delta|} r_{t_i} \cdot tf_{t_i,n} \tag{2.5}$$

donde:

- i es el i-ésimo término del contexto Δ.
- $|\Delta|$ es el número de términos del contexto.

- r_{t_i} es la relevancia del i-ésimo término de Δ (normalizado entre [0,1]).
- $tf_{t_i,n}$ es la frecuencia del término t_i en el nodo n, ver ecuación (2.6).
- $shared(n,\Delta)$ es el número de términos compartidos entre el nodo n y el contexto Δ.

Esta fórmula tiene dos partes diferentes que ayudan a balancear la *diversidad* y la *frecuencia*. La diversidad, medida por la cantidad de términos compartidos entre el nodo y el contexto, juega un papel importante para determinar la relevancia de los nodos. Por ejemplo, supongamos que solo consideramos la frecuencia, entonces el hecho de que un término aparezca muchas veces en el nodo puede ser suficiente para considerarlo relevante, ya que el valor de la frecuencia podría ser lo bastante alto. En la parte derecha de la fórmula se calcula el sumatorio de la frecuencia para cada término del contexto en el nodo. Con este valor tratamos de evaluar cuántas veces están contenidos los términos del contexto. Es necesario apuntar que la frecuencia de cada término en el nodo es pesada de acuerdo a la relevancia del término en el contexto, calculándose como sigue:

$$tf_{t,n} = \frac{num_{t,n} + sim_{t,n} + syn_{t,n}}{|n|} \tag{2.6}$$

donde:

- $num_{t,n}$ es el número de ocurrencias de t en n.
- $sim_{t,n}$ es el número de ocurrencias de los términos que son sintácticamente similares a t en n. Usamos SoftTFIDF para determinar el grafo de similitud entre los términos, igual que en contexto.
- $syn_{t,n}$ es el número de ocurrencias de los sinónimos de t que son incluidos en n.

El valor de $syn_{t,n}$ se calcula considerando las siguientes relaciones de la DBpedia, que almacenan los conceptos que son sinónimos: `dbpprop:alternativeNames`, `dbpprop:as` y `dbo:wikipageRedirects`. Por ejemplo, la entidad `dbo:Napoleon_I` contiene los valores *Napoleón Bonaparte I*, *Emperador francés* y *Rey de Italia* en la propiedad `dbpprop:alternativeNames`; y *Rey de Francia y Navarra* y *Rey de los franceses* en `dbpprop:as`. Los sinónimos recuperados en estas relaciones tienden a ser directamente procesables, aunque en algunos casos, como por ejemplo la propiedad `dbo:wikiPageRedirects`, no hay texto sino una URI que necesita ser procesada.

Selección de las relaciones semánticas

Para enriquecer la búsqueda y añadir un componente semántico al proceso de filtrado, expandimos el proceso de exploración incluyendo relaciones jerárquicas y taxonómicas de primer nivel en cada entidad (líneas 6-32 del algoritmo). Así, consideramos las siguientes propiedades de la DBpedia:

- `dct:subject`. Esta propiedad enlaza a las categorías de una entidad. Por ejemplo, la entidad *Homo sapiens* se clasifica en categorías como `Category:Tool_using_species` o `Category:Humans`. Las categorías son una alternativa para agrupar las entidades, pudiendo ser útiles para recuperar las entidades relacionadas. Teniendo esto en cuenta, nuestro algoritmo expande el espacio de búsqueda considerando también las categorías de los nodos hermanos (a través de la propiedad `is dcterms:subject of`).

- `skos:broader`. Esta propiedad se usa para establecer un enlace jerárquico entre dos categorías, indicando que una categoría es más general que otra. Usamos tanto `skos:broader` como su inversa, `is skos:broader of` para expandir el espacio de búsqueda. Por ejemplo, dos categorías generales de `Category:Humans` son `Category:Hominina` y `Category:Antropology`.

- `rdfs:type`. Usamos esta propiedad para establecer que una entidad es una instancia de una clase. Consideramos la tipología como si fuera un tipo de categoría, por ejemplo, recuperando las instancias del mismo tipo de la entidad.

- `rdfs:subClassOf`. Esta propiedad indica una relación de subsunción entre dos clases. La clase subsumida por otra se denomina subclase (relación `rdfs:subClassOf`), mientras que la clase que subsume a otra se denomina superclase (relación `is rdfs:subClassOf of`). Para cada sub o superclase recuperamos sus instancias que serán evaluadas como cualquier otra entidad del grafo.

2.4. Validación de resultados

Existen varias formas de medir la calidad de un sistema de anotación. Si enfocamos el problema de anotación como un problema de búsqueda, normalmente la efectividad de la recuperación se basa en la relevancia del documento con respecto a las necesidades del usuario.

Nombre del LO	Temática	Términos anotados
Antiguo Egipto	Historia	10
Mesopotamia	Historia	13
Paleolítico	Historia	10
Geografía ode Europa	Geografía	10
Paisaje de la Tierra	Geografía	7

Tabla 2.3: Conjunto de objetos de aprendizaje y el número de términos usados en la validación.

En el caso de la anotación, más concretamente en *EL*, hay que medir la correcta asociación entre los términos del documento y las entidades de la ontología. En esta tesis usamos como criterios para medir este aspecto la *precisión* y la *cobertura*. La precisión es el ratio entre las entidades relevantes recuperadas de la ontología y el total de entidades recuperadas, mientras que la cobertura es el ratio entre el número de entidades relevantes recuperadas y el total de entidades relevantes existentes en la ontología:

$$Precisión = \frac{|VP|}{|VP| + |FP|} \tag{2.7}$$

$$Cobertura = \frac{|VP|}{|VP| + |FN|} \tag{2.8}$$

donde VP (verdaderos positivos) es el conjunto de entidades recuperadas que son relevantes, FP (falsos positivos) es el conjunto de entidades recuperadas incorrectamente asignadas como relevantes; y FN (falsos negativos) son entidades relevantes que no fueron recuperadas.

Aunque *ADEGA* se podría haber aplicado a cualquier tipo de documentos, se ha validado en el dominio del aprendizaje electrónico, donde existe una necesidad de enriquecimiento de la información contenida en los objetos de aprendizaje (LO, del inglés *Learning Objects*). Concretamente hemos tratado con un tipo especial de LO: libros electrónicos de primaria[2], validando el algoritmo con un conjunto que cubre un cierto rango de temas de Geografía e Historia. Son documentos en formato XML que describe una estructura concreta del contenido, la figura 2.4 muestra alguno de los elementos de la estructura para un documento sobre el *Antiguo Egipto*. Las cajas negras representan los diferentes elementos de la estructura del LO; en concreto, la caja 1 identifica la sección del título; las cajas 2 y 6, la sección de contenido; la caja 3, un término marcado en negrita; la 4, un título de una imagen; y la caja 5, un enlace.

[2]Procedentes de una empresa orientada al aprendizaje electrónico

Cada una de estas cajas son consideradas *campos* del documento, utilizados para el cálculo del contexto (sección 2.2).

En la tabla 2.3 se listan los LO de cada uno de los temas utilizados. La columna derecha de la tabla muestra el número de términos relevantes usados en la validación y que son parte del contexto. Cada uno de estos términos fue sujeto de un proceso de validación por parte de expertos del dominio que consistió en (*i*) identificar la entidad de la DBpedia para enlazarla y (*ii*) filtrar el grafo relevante para esa entidad. Por ejemplo, uno de los diez términos relevantes del LO *Antiguo Egipto* es *Osiris*. En nuestro escenario, la validación consiste en comprobar que *Osiris* coincide con su correspondiente entidad en la DBpedia (en este caso, con `dbo:Osiris`), que se han identificado las relaciones relevantes en función del contexto y que las relaciones no relevantes se hayan descartado.

Un conjunto de expertos pedagogos identificó manualmente el grafo solución para cada uno de los términos relevantes de los objetos de aprendizaje. Las soluciones cubren dos niveles de profundidad de exploración, esto es, el nodo raíz del grafo solución (la entidad de partida) y los arcos hasta sus nodos nietos. La razón por la que hemos limitado las soluciones a dos niveles es porque cada nivel incrementa exponencialmente el número de nodos que tienen que ser comprobados, haciendo inviable el proceso manual de validación y construcción de la solución. Por ejemplo, si consideramos que cada entidad de la DBpedia tiene más de 40 relaciones (`dbo:Place` tiene 62 relaciones, y `dbo:Animal` tiene más de 49) el salto desde el segundo al tercer nivel implica procesar al menos 40^3 nodos adicionales. Aunque nuestro algoritmo no está limitado en esa profundidad, en la sección 2.4.1 mostramos que la profundidad de exploración es un factor importante en la calidad de los resultados independientemente de que las soluciones contienen nodos de nivel dos.

El proceso de comparación entre la solución identificada por los pedagogos y la solución obtenida por *ADEGA* está esbozada en la figura 2.5. El objetivo de este proceso es obtener los conjuntos de VP, FP, VN y FN para poder medir la precisión y la cobertura. La eficacia de nuestra solución se representa en la figura 2.6 y muestra unos resultados muy positivos, especialmente considerando que nuestra solución está desarrollada en el campo de la educación. Como se puede ver, nuestra solución tiene una precisión nada despreciable para un valor de cobertura cercano a 1,0 (entre 0,8 y 0,9), lo que garantiza que todo el contenido relacionado con un tema es recuperado. Además, esta precisión es considerablemente alta para valores intermedios de cobertura, lo que es importante si los estudiantes buscan un contenido concreto. Otro factor que determina la calidad de los resultados es el peso asignado a las relaciones. La

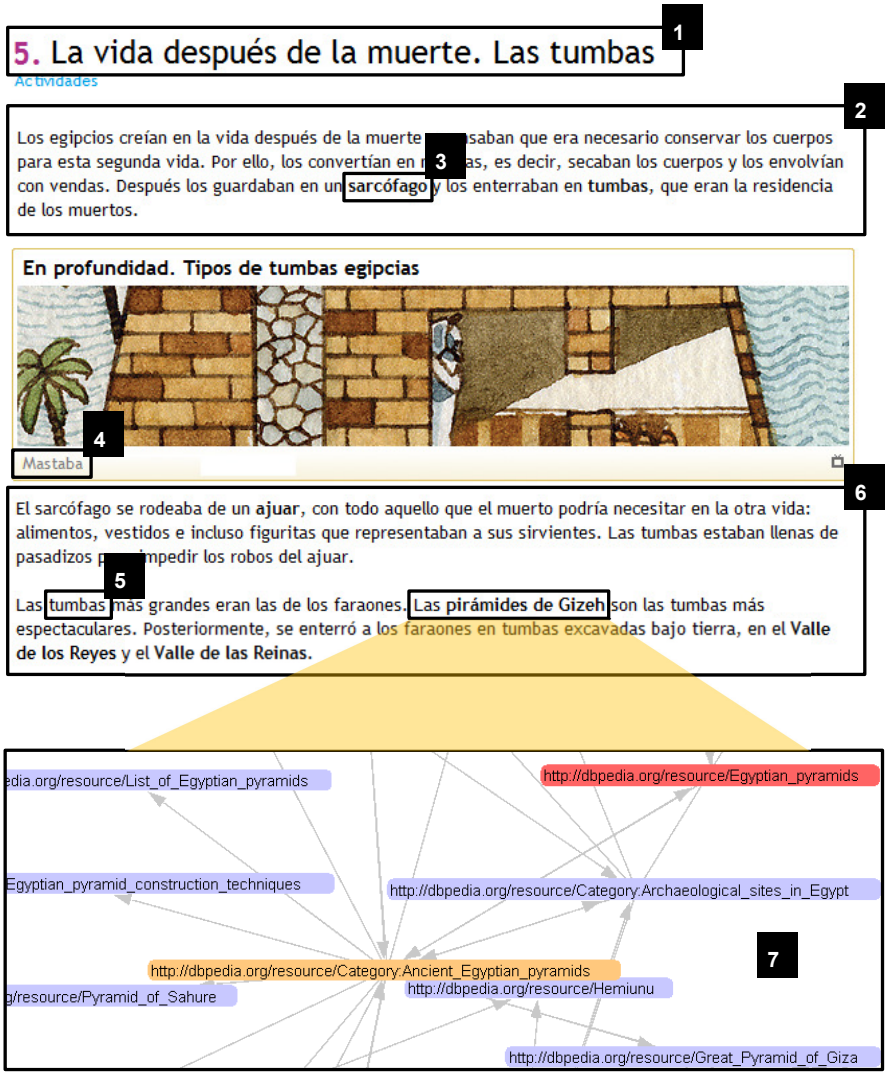

Figura 2.4: Ejemplo de un LO sobre el *Antiguo Egipto*, donde se han resaltado algunos campos estructurales del documento. La caja 7 representa un grafo semántico que anota el término *Pirámides de Gizeh* del LO.

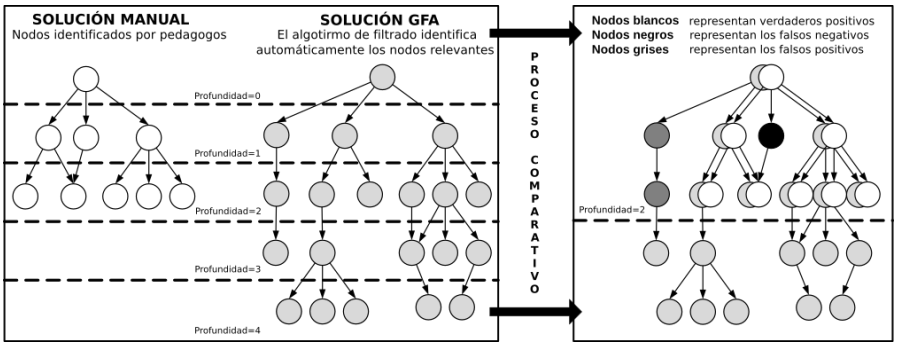

Figura 2.5: Método de comparativa entre los grafos definidos por los profesores y los grafos obtenidos porel algoritmo ADEGA.

sección 2.4.2 profundizará en los efectos de considerar diferentes configuraciones de pesos; y finalmente en la sección 2.4.3 compararemos nuestros resultados con otras aproximaciones similares disponibles en la literatura.

2.4.1. Límite de exploración del grafo

La figura 2.6 muestra la influencia del límite de exploración en la calidad de los resultados. En estas pruebas tuvimos en cuenta tres configuraciones diferentes según el nivel de exploración: nivel dos (igual que las soluciones proporcionadas por los profesores), tres y cuatro. Cada punto de la curva representa la media aritmética de la precisión de las 50 ejecuciones llevadas a cabo para una cobertura dada.

Las curvas muestran que hay una mejora tanto en precisión como en cobertura cuando la profundidad aumenta. El salto cualitativo se produce cuando la profundidad se incrementa del segundo al tercer nivel de exploración, e incluso con niveles bajos de cobertura como 0,3 o 0,4. Por ejemplo, hay una ganancia en precisión de un 10 % en el tercer nivel para un valor de cobertura de 0,3. La mejora es incluso mayor para niveles altos: la precisión incrementa de 0,2 a 0,6 para una cobertura de 0,6. En cierto sentido, este salto cualitativo se debe a que la relevancia de los últimos nodos viene determinada por las relaciones textuales del siguiente nivel; por lo tanto, si incrementamos la profundidad de exploración, el análisis se completa con la información disponible en los nodos del nivel siguiente, alcanzando una mejora significativa.

En el caso concreto de los LO analizados, cabe señalar que no hay una gran diferencia

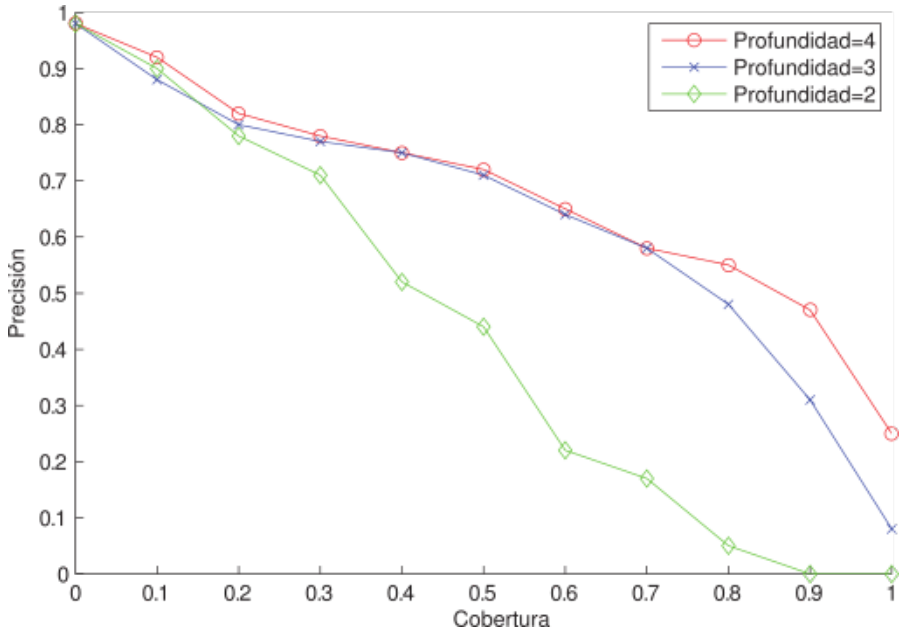

Figura 2.6: Curvas de precisión y cobertura para diferentes profundidades de exploración.

entre los niveles de exploración tres y cuatro. Solo se detecta una cierta mejora en niveles altos de cobertura, en concreto desde 0,8 en adelante. Este resultado es razonable ya que nosotros solo buscamos soluciones en nivel dos. La conclusión es que la exploración del nivel cuatro no proporciona información adicional para mejorar los resultados. Además, el coste/beneficio de aumentar el nivel de exploración es dudoso, ya que la exploración en este nivel implica analizar una gran cantidad de nodos. Manteniendo la asunción de que cada nodo tiene una media de 40 relaciones, implicaría una exploración de 40^4 nodos adicionales.

2.4.2. Peso de las relaciones

El peso de las relaciones es otro factor a tener en cuenta para determinar la relevancia de los nodos, tal como se indica en la ecuación 2.4. Sin embargo, ¿cuánto influye el peso de las relaciones en los resultados? Para contestar esta pregunta, hemos realizado una serie de pruebas midiendo el impacto de asignar una combinación específica de pesos a las relaciones.

Como decíamos en la sección 2.3, no todas las relaciones de la DBpedia son útiles para

Pesos			Cobertura										
A	**B**	**C**	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
1,0	0,7	0,7	1,0	,99	,89	,86	,80	,67	,50	,31	,10	0,0	0,0
1,0	0,7	1,0	1,0	,99	,88	,86	,80	,75	,51	,45	,14	0,0	0,0
1,0	0,8	1,0	1,0	,99	,88	,87	,79	,75	,49	,39	,13	0,0	0,0
1,0	0,9	1,0	1,0	,99	,88	,86	,79	,75	,48	,38	,12	0,0	0,0
1,0	1,0	1,0	1,0	,99	,88	,86	,79	,75	,48	,38	,12	0,0	0,0

Tabla 2.4: Las mejores 5 configuraciones de las relaciones `dbo:abstract`, `rdfs:label`, y `por defecto`. A representa a `dbo:abstract`, B a `rdfs:label`, C para el resto de relaciones.

determinar si un nodo es o no relevante. En este trabajo hemos limitado el análisis a aquellas relaciones que consideramos más influyentes y que pueden ser encontradas en la mayoría de las entidades (ver tabla 2.2). Todas ellas toman valores en el intervalo $[0,1]$, donde 0 indica que la relación no es relevante; y 1 que es muy importante. Las primeras dos relaciones son relaciones hacia texto, proporcionan el nombre y la descripción de una entidad, mientras que las otras seis son consideradas relaciones *semánticas*, en el sentido de que pueden ser usadas para relacionar nodos a través de su tipología y categorías. Otras relaciones fueron directamente descartadas, todas aquellas cuyo rango son números o tipos de datos a los que no se les puede aplicar ningún análisis sintáctico o semántico. Finalmente, el resto de relaciones son consideradas instancias de la misma relación, denotadas como relaciones *por defecto*.

Para contestar nuestra pregunta inicial, hemos analizado el comportamiento del algoritmo de filtrado con diferentes configuraciones de pesos para dos niveles de profundidad. Teniendo en cuenta que realizar un conjunto de pruebas exhaustivo implicaría ejecutar un mínimo de 10^{11} experimentos, considerando solo 10 posibles valores para cada tipo de relación; hemos optado por afrontar el problema en tres pasos secuenciales:

1. En el primero, se determina la mejor configuración para tres relaciones que basadas en nuestra experiencia tienen más influencia en los resultados. En concreto consideramos `dbo:abstract`, `rdfs:label` y la relación `por defecto`. La tabla 2.4 muestra los resultados de esta primera prueba. Las primeras tres columnas representan el peso asignado a cada relación, mientras que las últimas diez columnas indican la precisión obtenida para un valor concreto de cobertura. Como se puede ver en la figura 2.7, las mejores soluciones se obtienen cuando el valor de `dbo:abstract` está entre el ratio [0,8;1,0]; el de `rdfs:label` entre [0,5;0,8] y el de la relación `por defecto`

Pesos				Cobertura										
D	**E**	**F**	**G**	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
0,9	1,0	0,6	1,0	1,00	0,99	0,89	0,87	0,81	0,77	0,53	0,46	0,14	0,0	0,0
0,9	0,9	0,9	0,9	1,00	0,99	0,89	0,87	0,81	0,77	0,52	0,46	0,14	0,0	0,0
0,9	1,0	0,9	1,0	1,00	0,99	0,89	0,87	0,80	0,77	0,53	0,46	0,14	0,0	0,0
1,0	0,9	1,0	0,9	1,00	0,99	0,89	0,87	0,80	0,77	0,53	0,48	0,14	0,0	0,0
1,0	1,0	1,0	1,0	1,00	0,99	0,89	0,87	0,80	0,77	0,54	0,48	0,14	0,0	0,0

Tabla 2.5: Las 5 mejores configuraciones para las relaciones basadas en el vocabulario SKOS. D representa a `dct:subject`, E a `is dct:subject of`, F a `skos:broader`, y G a `is skos:broader of`.

entre $[0{,}8; 1{,}0]$. Es decir, se muestra que estas relaciones tienen una gran relevancia. Las configuraciones son bastante razonables teniendo en cuenta nuestra asunción de que asignándoles un peso alto, mejorarían los resultados. La fila con la mejor configuración está resaltada y en este caso, los valores asignados para `dbo:abstract`, `rdfs:label` y la relación `por defecto` son 1,0, 0,7 y 1,0 respectivamente. Aunque la diferencia con otras configuraciones es mínima, hemos seleccionado esta configuración porque tiene la mayor precisión en los valores más altos de cobertura, y está entre las mejores configuraciones para niveles bajos de cobertura.

2. En el segundo, se determina la mejor configuración para las relaciones basadas en el vocabulario SKOS. En este caso, pesamos las relaciones `dct:subject`, `skos:broader`, `is dcterms:subject of` y `is skos:broader of`, pero manteniendo la mejor configuración para las relaciones del paso previo (`dbo:abstract`, `rdfs:label` y la relación `por defecto`). La tabla 2.5 muestra algunos de los valores obtenidos para cada configuración. La precisión varía menos que en el caso anterior y para valores bajos de cobertura no hay diferencia entre las configuraciones (por ejemplo, para una cobertura de 0,8 sólo hay una diferencia de 0,6 en la precisión entre la mejor y la peor configuración). La diferencia entre asignar el peso de estas relaciones a 0,6 o 1,0 es mínima y solamente es apreciable en valores bajos de cobertura. Además, se puede ver que hay dos soluciones no dominantes, aunque en esta tesis consideremos el caso $[0{,}9; 1{,}0; 0{,}6; 1{,}0]$ como la mejor configuración porque tiene mayor precisión en valores altos de cobertura.

3. Por último, se determina la mejor configuración para las relaciones "es-un" ("*is-a*" en inglés). Este tipo de relaciones se usan para expandir la consulta con los parientes del

Pesos				Cobertura										
H	**I**	**J**	**K**	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
0,6	0,6	0,6	0,6	1,00	0,99	0,89	0,87	0,80	0,77	0,53	0,48	0,14	0,0	0,0
0,6	1,0	0,6	0,9	1,00	0,99	0,89	0,87	0,79	0,76	0,49	0,40	0,12	0,0	0,0
0,6	1,0	0,6	1,0	1,00	0,99	0,89	0,88	0,80	0,76	0,50	0,41	0,13	0,0	0,0
0,9	0,6	0,9	0,6	1,00	0,99	0,89	0,87	0,80	0,77	0,53	0,48	0,14	0,0	0,0
1,0	1,0	1,0	1,0	1,00	0,99	0,89	0,87	0,80	0,77	0,53	0,48	0,14	0,0	0,0

Tabla 2.6: Las 5 mejores configuraciones para las relaciones "es-un". H representa a `rdf:type`, I a `is rdf:type of`, J a `rdfs:subClassOf`, y K a `is rdfs:subClassOf Of`.

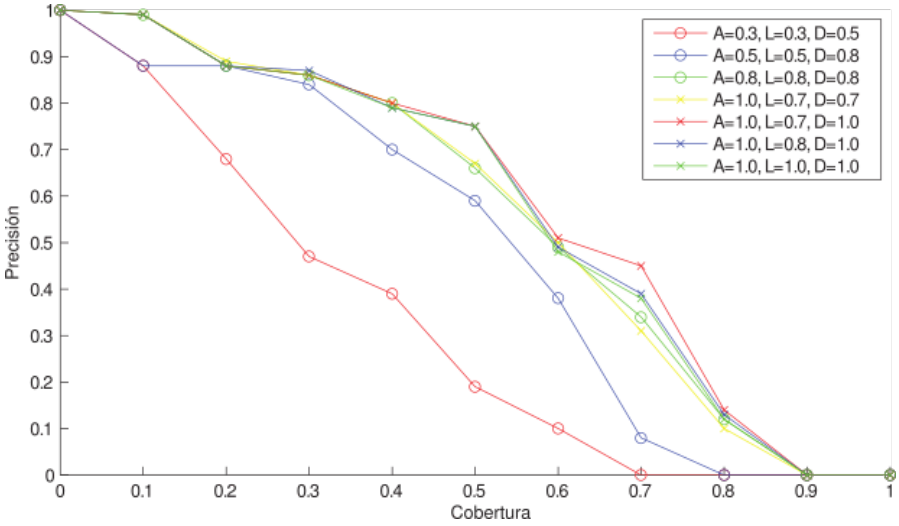

Figura 2.7: Curvas de precisión y cobertura para diferentes configuraciones de pesos de las relaciones `dbpedia-owl:abstract`, `rdfs:label` y la relación `por defecto`.

nodo examinado y proporcionar nodos interesantes relacionados con la temática del documento. Aun así, en la tabla 2.6 se puede apreciar la influencia mínima de estas relaciones, ya que no hay diferencia en términos de precisión para pesos entre 0,6 y 1,0. Aunque estas pruebas fueron hechas para la mejor configuración del paso 2, otros experimentos nos sugieren que esta conclusión también se aplica a cualquier otra configuración. En nuestra opinión, este comportamiento se debe a que existen muchos nodos hijos y padres que no están relacionados con el tema del documento, reduciendo así la contribución de las relaciones.

Aunque no puede apreciarse a partir de las tablas anteriores, nuestra experimentación muestra que la configuración final se ajusta mejor a algunos objetos de aprendizaje que a otros (por ejemplo, los documentos de historia se comportan mejor que los de geografía). Además, hemos identificado casos donde una relación específica tiene un efecto importante en cierto tipo de nodos. Este hecho sugiere que el peso de las relaciones también puede depender de la tipología de la entidad a la que se aplica. Por lo tanto, creemos que probablemente sea necesario ajustar la configuración del peso en cada tipo, de modo que con ello se mejorarían los resultados de la exploración global. Sin embargo, este análisis está fuera del alcance de este trabajo.

2.4.3. Comparativa con otras aproximaciones

En esta sección compararemos nuestra propuesta con *RelFinder* [3] [42]. La principal funcionalidad de RelFinder es extraer un grafo compuesto por nodos que forman caminos conectando dos o más entidades de partida, pudiéndose establecer el nivel de profundidad como el número de nodos entre las dos entidades (la longitud del camino). Además, *RelFinder* devuelve relaciones que se ajustan a una consulta, lo que permite personalizar las relaciones a utilizar en la exploración. La diferencia con nuestro algoritmo, es que *RelFinder* no realiza el proceso de identificación y enlazado de entidades inicial. Tampoco lleva a cabo ningún tipo de filtrado del grafo una vez creado, ni utiliza las entidades de partida para obtener un mejor conjunto de caminos.

Teniendo esto en cuenta, comparamos los grafos RDF de la aplicación y los de nuestra propuesta con la solución indicada por los pedagogos. Esta comparativa se basa en la métrica *medida-F* (conocida como *F-measure* o *F-score* en inglés):

$$F_\beta = (1+\beta^2) \cdot \frac{precisión \cdot cobertura}{(\beta^2 \cdot precisión) + cobertura} \tag{2.9}$$

Como se puede apreciar en la ecuación 2.9, la fórmula de la *medida-F* considera la precisión y la cobertura para calcular la *calificación* del test. En nuestra comparativa usamos la versión balanceada de esta métrica, que establece $\beta = 1$, y puede ser interpretada como la media armónica entre la precisión y la cobertura.

La tabla 2.7 muestra los resultados de la comparativa, donde cada columna representa a un LO del que se extrajeron los términos del contexto. Para cada LO, la comparativa se ha llevado a cabo como sigue:

[3] El demostrador está disponible en la URL http://www.visualdataweb.org/relfinder/relfinder.php

RelFinder	LO_1	LO_2	LO_3	LO_4	LO_5
P	0.20	0.12	0.27	0.09	0.46
R	0.15	0.04	0.12	0.04	0.09
F_1	0.17	0.06	0.17	0.06	0.15
ADEGA(δ_d=2)					
P	**0.58**	**0.79**	**0.51**	**0.65**	**0.66**
R	**0.42**	**0.25**	**0.23**	**0.28**	**0.13**
F_1	**0.49**	**0.38**	**0.32**	**0.39**	**0.22**

Tabla 2.7: Comparación de *RelFinder* y ADEGA (dos niveles de profundidad). LO_1=Paleolítico, LO_2=Paisaje de la Tierra, LO_3=Mesopotamia, LO_4=Geografía de Europa y España, LO_5=Antiguo Egipto. P, R, y F_1 representan a la precisión, cobertura y medida-F_1, respectivamente. Las celdas con la mejor configuración están resaltadas en cada columna.

- Se han introducido en *RelFinder* los 20 primeros términos más relevantes del contexto (configurado para generar caminos de longitud 2).

- ADEGA recupera grafos para los 10 primeros términos del contexto, pero considera los 20 más relevantes para que formen parte del contexto.

- Obtenemos *para el mismo número de soluciones* los valores de cobertura y precisión de *RelFinder* y ADEGA. Esto significa que *RelFinder* fue configurado para tener en cuenta las mismas relaciones de la DBpedia que ADEGA (ya que por defecto *RelFinder* descarta algunas relaciones como `rdfs:type` y `rdfs:subClassOf`). En esta configuración se han considerado todas las relaciones menos `dbo:wikiPageWikiLink`, porque conectan a una gran cantidad de nodos que deberían ser analizados por los algoritmos, además de no proporcionar información semántica sobre el nodo.

En la tabla 2.7 podemos ver que nuestro algoritmo de filtrado mejora la propuesta de *RelFinder* en todas las pruebas. Incluso para dos niveles de exploración, nuestro peor caso, ADEGA obtiene mejores resultados. La mayor diferencia se aprecia en LO_4 donde *RelFinder* y ADEGA obtienen valores de 0,06 y 0,39 respectivamente. Este mismo ratio se mantiene en la mayoría de los objetos de aprendizaje utilizados en los textos, aunque disminuye un poco en LO_3 y LO_5. De hecho, si comparamos la media de la medida-F_1 para los cinco documentos, *RelFinder* obtiene 0,13 mientras que ADEGA consigue 0,36.

Aunque esta comparativa se ha realizado con contextos extensos, solo se han tenido en cuenta en el experimento cinco objetos de aprendizaje. Con el fin de determinar si podemos extrapolar estos resultados tanto para un número mayor de documentos como para diferentes tipos de textos, hemos realizado otro experimento con 40 objetos de aprendizaje del repositorio de Universia. Los resultados pueden consultarse en el capítulo 5, donde se explica el caso de aplicación real de anotación semántica sobre este repositorio.

2.5. Conclusiones

En este capítulo presentamos ADEGA, una novedosa aproximación a la anotación semántica que, al contrario de las propuestas tradicionales, anota los términos relevantes del documento con (sub)grafos extraídos de repositorios de datos enlazados. Estos grafos, además de ser claves en la mejora el proceso de anotación, también enriquecen semánticamente el documento, ya que aportan conceptos que originalmente no aparecen en el texto sino que se obtienen de las relaciones que parten de los términos originales del documento. Esta característica, unida al algoritmo de búsqueda utilizado (en profundidad), hace que la precisión y cobertura de ADEGA sea muy dependiente del nivel de exploración. Este aspecto es todavía más crítico si consideramos el tiempo de computación, ya que el tiempo necesario para anotar un término crece exponencialmente por cada nivel adicional explorado. Sin embargo, tal y como se detalló en la experimentación, la precisión y cobertura no mejoran en esa misma medida con cada nivel adicional de exploración. En este sentido, las pruebas confirmaron que a partir de tres niveles, la ganancia obtenida no compensa el tiempo invertido en explorar el grafo.

Otro de los objetivos de este capítulo ha sido comparar la solución propuesta con otras similares disponibles en la literatura, es decir, con otras soluciones que anotan un término del documento con un grafo. Sin embargo, en la práctica, ADEGA únicamente pudo compararse con *RelFinder*, ya que la mayoría de las aproximaciones anotan términos con una única entidad. Este detalle es sumamente importante ya que en la práctica implica que las entidades del grafo que no representan un término relevante del documento no se consideran y, por ello, estas soluciones nunca podrían obtener niveles aceptables de precisión. En cambio, *RelFinder* dispone de un mecanismo de anotación cuyo objetivo es la creación de un grafo conexo entre las entidades que representan los términos relevantes del documento a anotar. El resultado de la comparativa (ver tabla 2.7) demuestra que ADEGA mejora con claridad a *RelFinder* tanto en precisión, cobertura como medida-F.

Tal y como se ha descrito en este capítulo, los resultados obtenidos por ADEGA en términos de precisión y cobertura mejoran el estado del arte de la anotación semántica basada en grafos. Sin embargo, este tipo de anotación es computacionalmente costosa, tanto en tiempo como en recursos computacionales. Por esta razón, en el siguiente capítulo se detallará una nueva versión de ADEGA, pensada para entornos de altas prestaciones, y cuyo objetivo es facilitar la anotación de grandes volúmenes de datos en tiempo real.

Capítulo 3

Paralelización de la anotación semántica en ADEGA

3.1. Introducción

Tal como se ha comentado en el capítulo anterior, desde un punto de vista de anotación semántica, el algoritmo ADEGA mejora el estado del arte en términos de precisión y cobertura. Al contrario que las propuestas clásicas, la anotación se realiza a través de grafos, lo cual permite el enriquecimiento de los contenidos y como consecuencia directa puede mejorar las búsquedas futuras almacenando estos metadatos; sin embargo, con un coste computacional añadido. En el capítulo anterior se validó el sistema aplicando ADEGA sobre un pequeño conjunto de objetos de aprendizaje, concretamente un conjunto de libros digitales cuya temática era Historia y Geografía. Sin embargo, uno de los aspectos que se ha revelado en esta validación es el pobre rendimiento en relación al tiempo transcurrido en obtener un grafo para cada término y, por extensión, para cada libro digital. Con el objetivo de valorar esta eficiencia, decidimos aplicar el algoritmo ADEGA a la anotación y enriquecimiento del repositorio de objetos de aprendizaje de Universia [3]. En el capítulo 5 se detalla de qué modo se ha realizado el enriquecimiento y anotación de este repositorio.

El repositorio de Universia almacena 213 colecciones de objetos de aprendizaje pertenecientes a diferentes campos científicos. En total estas colecciones consisten en más de 15 millones de objetos de aprendizaje que sirven como recursos académicos. Los metadatos de estos recursos fueron inicialmente descritos conforme el estándar IEEE LOM [23] y cate-

gorizados utilizando la nomenclatura de UNESCO [103]. No obstante, un análisis detallado del repositorio evidenció una serie de problemas que dificultaban enormemente la búsqueda y recuperación de estos recursos, entre otros: los metadatos de algunos de ellos estaban incompletos y una gran cantidad estaban incorrectamente categorizados. Ante esta situación, se propuso categorizar de nuevo las colecciones de objetos de aprendizaje, utilizando las categorías de la DBpedia como alternativa a la nomenclatura de UNESCO, aplicando ADEGA al repositorio Universia. Ante tal cantidad de datos procedentes del repositorio, se concluyó que una anotación directa no era viable: una estimación preliminar determinó que se necesitarían más de 1.640 años de CPU para crear grafos de nivel tres; y más de 25.000 años para grafos de profundidad cuatro. Obviamente, con estos requisitos de computación un ordenador personal o incluso un pequeño conjunto de recursos computacionales no podrían resolver en un tiempo asumible el proceso de anotación para la gran cantidad de objetos.

En este capítulo, describimos cómo se ha utilizado una infraestructura de integración de recursos computacionales [29] para resolver el problema de anotar semánticamente el repositorio de Universia, con le objetivo de comprobar si con esa solución se resuelve el problema de eficiencia de ADEGA. Además, se trata de una experiencia práctica y real de combinar técnicas de anotación semántica y computación distribuida en el contexto de la anotación a gran escala.

3.2. Anotación semántica del repositorio de Universia

Nuestro sistema de anotación hace uso de las categorías de la DBpedia para mejorar la clasificación de los objetos de aprendizaje de Universia a través de un proceso de filtrado de grafos, donde se seleccionan las categorías más adecuadas para clasificar al objeto y se anotan los términos relevantes del contenido. Una vez se seleccionan las categorías y las entidades, se completa la información de los metadatos. ADEGA nos proporciona la forma de enriquecer el contenido del objeto y mejorar su clasificación. En el capítulo 5 se detalla cómo se ha realizado el proceso de enriquecimiento de objetos de aprendizaje.

A pesar de las ventajas del algoritmo para realizar este proceso de anotación, el coste de la exploración para obtener un grafo de la DBpedia que anota un término relevante es muy costoso: $O(b^n)$, donde b es el factor de ramificación (número de relaciones entre nodos) y n es el nivel más profundo de exploración. Por lo tanto, antes de usar ADEGA, hemos comprobado la viabilidad de esta aproximación ya que, como se ha comentado anteriormente, el reposi-

Medida	Valor
Media de nodos visitados	248.035,02
Media de nodos descartados	199.461,73
Media de nodos procesados	48.573,29
Media de nodos de datos primitivos procesados	44.165,53
Media de nodos entidad procesados	4.407,76
Media de consultas SPARQL realizadas	22.882,64
Tiempo medio por consulta en ms.	9,91
Tiempo medio de ADEGA (profundidad = 3) en ms.	376.414,17

Tabla 3.1: Indicadores de experimentación de la versión secuencial de ADEGA al anotar 10.000 objetos de aprendizaje.

torio de Universia está compuesto por 213 colecciones de objetos y un total de 15.750.979 de recursos académicos. La primera medida para disminuir tiempos fue limitar el nivel de exploración. Como se ha comentado en el capítulo anterior, considerar más de tres niveles de exploración no mejora significativamente la precisión y la cobertura.

Hay que recalcar que incluso limitando el nivel de profundidad, este tiempo es considerable y hace inviable el uso de ADEGA en aplicaciones reales. En la tabla 3.1 se incluyen una serie de medidas sobre el rendimiento de ADEGA al anotar 10.000 objetos de aprendizaje seleccionados aleatoriamente. Durante este proceso, fueron identificados 79.628 términos clave y cada uno de ellos fue anotado con un grafo limitado a tres niveles de exploración. Como se puede ver en la tabla, cada grafo recuperado por el algoritmo se crea en un tiempo medio de 376.000 ms, más de 6 minutos. Considerando que cada LO es caracterizado con al menos 10 términos, en la versión secuencial de ADEGA se necesitarían 56 minutos para anotar un objeto. Si extrapolamos este número a los más de 15 millones de objetos de aprendizaje, se requerirían más de 1.640 años para anotar el repositorio completo de Universia.

Un análisis más profundo muestra que ADEGA visita una media de 248.035 nodos por ejecución y descarta la mayoría de ellos, concretamente el 80 %. Este es un indicativo importante, que enfatiza la necesidad del proceso de filtrado para evitar incorporar a la solución información no relevante de acuerdo al contexto del objeto. En concreto, se analizan una media de 44.165 nodos de datos primitivos usando técnicas de procesamiento de lenguaje natural. El resto de nodos procesados, 4.407, son entidades que requieren 22.882 consultas SPARQL para recuperar sus nodos hijos. En términos de coste del algoritmo, si multiplicamos el número de consultas realizadas por su tiempo medio (9,91 ms), podemos deducir que el 67 % del

tiempo de cómputo de ADEGA se dedica al acceso y consulta de la DBpedia.

En resumen, los problemas de ADEGA están relacionados con el hecho de que la creación del grafo es una tarea costosa y requiere mucho tiempo de ejecución. Además, el tiempo utilizado se incrementa exponencialmente con cada nivel de exploración; ya que el número de consultas SPARQL a realizar es muy elevado y el procesamiento de los nodos de datos primitivos también consume una gran cantidad de tiempo. Estos problemas se acentúan cuando es necesario anotar semánticamente un gran número de términos. Concretamente, es necesario procesar más de 15 millones de objetos de aprendizaje, lo que representan 150 millones de términos.

3.3. Paralelización mediante una infraestructura de integración de recursos

La versión original del algoritmo ADEGA no está optimizada para anotar eficientemente grandes colecciones de datos. Se puede proporcionar una solución a este problema (*i*) programando una versión paralela y más eficiente del algoritmo; e (*ii*) integrando recursos computacionales con el fin de conseguir todas las unidades de procesamiento disponible para ejecutarlo en un entorno de computación distribuida. Para ello, se va a reutilizar una infraestructura de gestión de recursos, implementada como parte de un trabajo previo [29]. A continuación, se describe brevemente la arquitectura y las características de esta infraestructura y los recursos que han sido integrados. Posteriormente, presentamos una visión de alto nivel de la aplicación paralela programada.

3.3.1. Descripción de la infraestructura

En [29] se presenta una infraestructura orientada a servicios para el despliegue y la ejecución de flujos de trabajo científicos. Esta infraestructura integra y gestiona de forma transparente un amplio conjunto de recursos de computación existentes en la Universidad de Zaragoza (*clúster*, recursos en la nube y de tipo *grid*) y los ofrece al usuario como un único y potente entorno de ejecución. La infraestructura está orientada a la resolución de problemas que requieren una computación intensiva y que manejan grandes volúmenes de datos. Por este motivo, hemos decidido utilizarlo con el propósito de anotar semánticamente el repositorio de Universia.

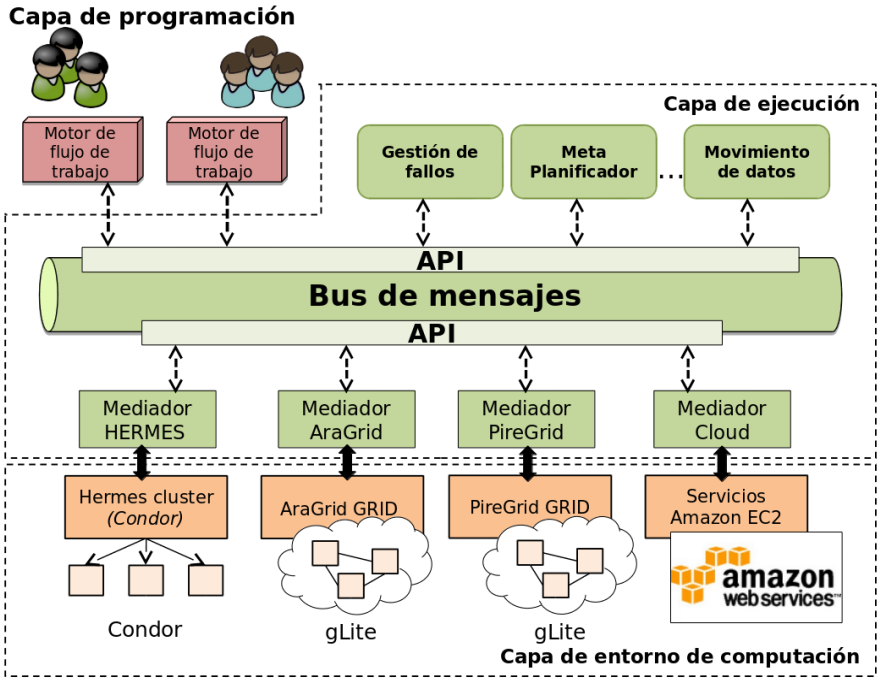

Figura 3.1: Diseño de la arquitectura de una infraestructura flexible para un desarrollo de flujos de trabajo en entornos heterogéneos

La arquitectura de la infraestructura está basada en un bus de mensajes inspirado en el modelo de coordinación *Linda* [16]. Este componente publica a través de su interfaz un conjunto de operaciones para el intercambio asíncrono de mensajes codificados como tuplas *Linda*. El resto de componentes ofrece su funcionalidad a través del bus compartido y colaboran entre sí por medio del intercambio de mensajes. Este modelo de integración permite reducir el acoplamiento entre los componentes del sistema, posibilita modificar la arquitectura de forma dinámica, favorece la escalabilidad y distribución de la solución, y facilita la utilización de múltiples patrones de interacción. Por lo tanto, el bus facilita que el modelo de integración de la solución sea altamente flexible [30].

La figura 3.1 muestra el diseño de la arquitectura de la infraestructura, formado por tres capas. En la parte superior, la *capa de programación o interfaz de usuario* está compuesta por los entornos de programación de flujos de trabajo que pueden ser usados para desarrollar

las aplicaciones ejecutadas sobre la infraestructura (por ejemplo, Taverna [45], Kepler [62], etc.). Estas aplicaciones deben ser orientadas a servicios para acceder a la funcionalidad de cómputo ofrecida por la infraestructura. Es decir, por cada una de las etapas de ejecución (tareas) de los flujos de trabajo, debe haber un servicio web que ejecuta la funcionalidad de dicha etapa. La capa intermedia, llamada *capa de ejecución*, es la responsable de gestionar el despliegue de las aplicaciones. Los componentes que constituyen esta capa están integrados en torno al bus de mensajes compartido. Existen dos tipos de componentes: los componentes de gestión y los mediadores. Los primeros son los responsables de proporcionar la funcionalidad necesaria para gestionar el ciclo de vida de las aplicaciones: meta-planificación, tolerancia a fallos, movimiento de datos, etc. Por su parte, los mediadores facilitan el acceso a los recursos de computación integrados en la infraestructura. Cada mediador es capaz de interaccionar con una infraestructura de computación concreta (por ejemplo, Amazon EC2), encapsulando sus detalles específicos, con el objetivo de enviar tareas para su ejecución, monitorizar el estado de las mismas, recuperar los resultados finales, etc. Por último, la *capa compuesta por las infraestructuras de computación* integra los recursos de computación heterogéneos accesibles a través de la infraestructura. Para el desarrollo de este trabajo se han integrado:

- El *clúster* HERMES del Instituto de Investigación en Ingeniería de Aragón (I3A)[1], con 1.308 procesadores y 2,92 TB de RAM.
- Los *grid* AraGrid[2] y PireGrid[3] del Instituto de Biocomputación y Física de Sistemas Complejos (BIFI)[4], con 1.728 procesadores y 3,38 TB de RAM para el primero; y 600 procesadores y 2 TB de RAM para el segundo.
- La infraestructura en la nube Amazon EC2[5]. Ofrece una visión de recursos infinitos en la que el usuario paga por los recursos que solicita. En este caso, se han utilizado 4 instancias de computación *m1.medium* para gestionar las tareas fallidas.

La infraestructura de integración permite acceder a estos recursos como un todo, sin que un usuario sea consciente de los diferentes paradigmas de computación integrados o de las distintas tecnologías lógicas de intercambio de información utilizadas para gestionar estos

[1] http://i3a.unizar.es
[2] http://aragrid.es
[3] http://www.piregrid.eu
[4] http://bifi.es
[5] http://aws.amazon.com/ec2/

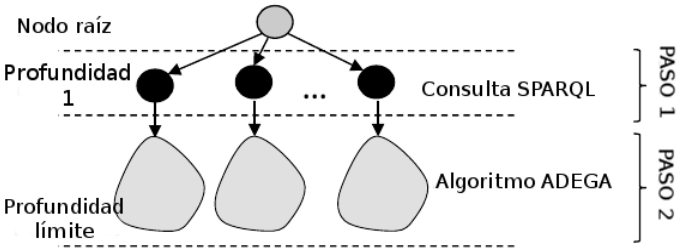

Figura 3.2: Versión paralelizada, alternativa a la versión secuencial de ADEGA

recursos (HERMES utiliza la tecnología de intercambio de información HTCondor, AraGrid y PireGrid utilizan gLite [58]).

3.3.2. Aplicación paralela para la anotación de objetos

Nuestro objetivo es proponer una solución alternativa para este tipo de casos, mucho más apropiada en términos de tiempo de ejecución; por lo que hemos definido una versión paralela del algoritmo para calcular el grafo de un término. Esta nueva versión está basada en el algoritmo original y en técnicas de programación paralela con el objetivo de desplegar y ejecutar el algoritmo en la infraestructura de integración descrita en la sección anterior. Esta infraestructura está especialmente ideada para ejecutar aplicaciones programadas utilizando entornos y lenguajes orientados a flujos de trabajo. En esta propuesta, el programador sólo debe preocuparse de describir el flujo de control y los datos de la aplicación, y no de especificar los recursos concretos de ejecución para cada una de las tareas del flujo. Internamente, una vez desplegada la aplicación, la infraestructura se encarga de determinar en qué recurso se ejecuta cada una de las tareas pendientes utilizando un meta-planificador basado en técnicas de simulación.

En la figura 3.2 se representa el proceso para calcular el grafo de un término en la versión paralela del algoritmo ADEGA. En esta nueva versión, el grafo se calcula en dos pasos: inicialmente, se calcula el nodo raíz del grafo (la entidad de la que parte la exploración) y se obtienen sus nodos hijos de primer nivel mediante una consulta SPARQL hacia la DBpedia; posteriormente usando la versión secuencial de ADEGA, se calcula en paralelo el grafo correspondiente a cada nodo hijo. Los subgrafos calculados en el segundo paso deben integrarse con los nodos del primer paso, con el propósito de obtener el grafo final del término a anotar.

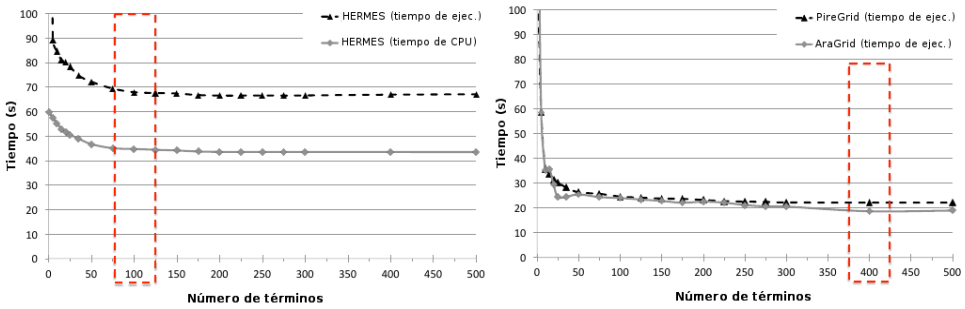

Figura 3.3: (a) Media de tiempo de ejecución y CPU para la infraestructura HERMES; (b) Media de tiempo de ejecución para las infraestructuras AraGrid y PireGrid

Intuitivamente, la nueva versión del algoritmo trata de calcular en paralelo tanto los grafos de las ramas del término raíz como los grafos de un conjunto de términos.

Antes de programar la aplicación del flujo de control, el primer paso es determinar el concepto de tarea en el problema concreto a resolver. En nuestro caso, una tarea debería ser responsable de la creación del grafo de uno o más términos tal como se explicó en el párrafo anterior. En este contexto, el número de términos anotados por cada tarea recibe el nombre de *granularidad*. Dado que las infraestructuras de computación integradas presentan diferentes configuraciones y recursos de cómputo, el siguiente paso es definir la granularidad más adecuada para cada una (un tamaño de tarea inadecuado puede provocar retardos significativos en la ejecución de la misma como consecuencia de largas esperas o expulsiones). Como parte de este trabajo, se ha determinado experimentalmente que la granularidad óptima para HERMES es 100 términos por tarea y para AraGrid y PireGrid, 400 términos por tarea. En la figura 3.3 se representan los términos por tarea frente al tiempo de ejecución y de CPU en cada uno, pudiéndose comprobar que la curva se estanca en 100 y 400 términos respectivamente.

Por otro lado, también se han adoptado ciertas decisiones iniciales relativas a la gestión de los datos con los que se pretenden favorecer la ejecución de la aplicación. En primer lugar, se ha desplegado previamente una copia de la DBpedia en cada nodo de computación, lo cual es posible porque el algoritmo usa la DBpedia en modo lectura. En segundo lugar, se han copiado los datos de entrada (los términos a anotar semánticamente) localmente en cada infraestructura antes del comienzo de la aplicación (el tamaño de estos datos es aproximadamente de 30 GB). De la misma forma, se han copiado los grafos generados como resultado

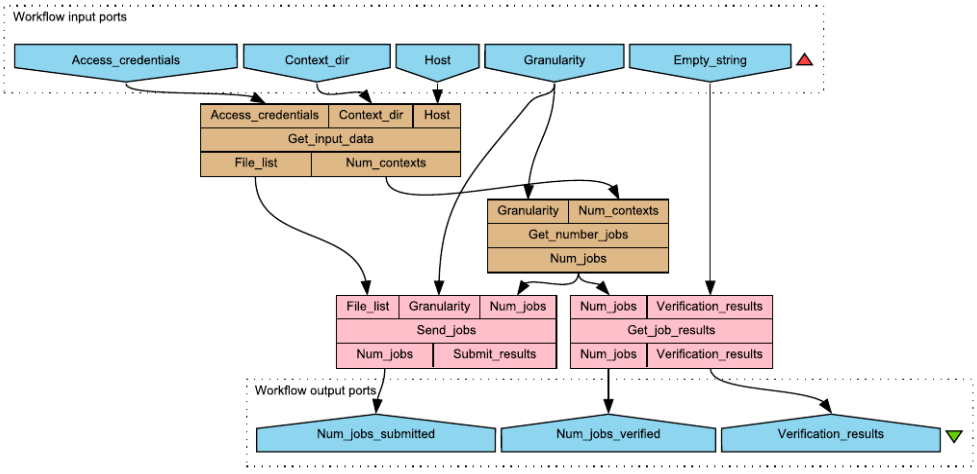

Figura 3.4: Implementación de un flujo de trabajo usando Taverna

desde la infraestructura donde fueron calculados a un sistema de almacenamiento externo una vez finalizada la aplicación (más de 120 GB de datos).

La figura 3.4 muestra la aplicación programada utilizando la sintaxis de alto nivel de Taverna por simplicidad y legibilidad (la aplicación real fue programada en Redes de Referencia utilizando la herramienta *Renew* [57]). Desde un punto de vista de alto nivel, la aplicación recibe los términos que deben ser anotados y genera como resultado los correspondientes grafos semánticos. Como datos de entrada se especifica la ubicación de los términos a anotar (*Host* y *Context_dir*) y las credenciales de acceso necesarias (*Access_credentials*). Además, es necesario indicar la granularidad óptima definida para cada infraestructura de computación (*Granularity*). Por otro lado, los resultados de la ejecución son los grafos almacenados localmente en las infraestructuras y una serie de parámetros que determinan si las tareas han sido ejecutadas correctamente (*Num_jobs_verified*, *Verification_results*, y *Num_jobs_ submitted*). La aplicación comienza ejecutando la tarea *Get_input_data* que recupera el listado de términos a anotar. El resultado es un conjunto de ficheros que contienen esta información y el número de contextos totales que deben ser anotados (*File_list* y *Num_contexts*). Una vez los términos y sus contextos son conocidos, la tarea *Get_number_jobs* calcula el número de tareas a ejecutar en base a la granularidad definida para cada infraestructura (*Num_jobs*). El sub-flujo de trabajo *Send_jobs* es el responsable de configurar y ejecutar la paralelización de

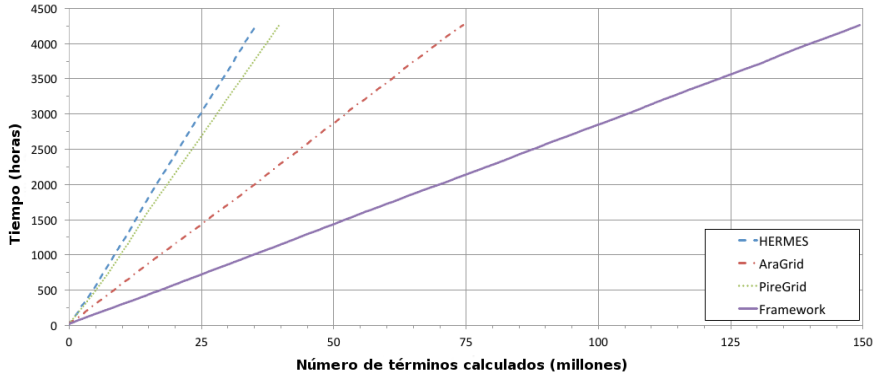

Figura 3.5: Número de términos enviados a cada infraestructura de computación y número total de términos enviados al sistema

las tareas de anotación. Internamente, crea tantas tareas como sea necesario para completar la anotación semántica de los términos de entrada, e invoca la interfaz de la infraestructura para su envío. La ejecución de una tarea consiste en la creación de los grafos semánticos que anotan cada término de entrada. Posteriormente, la propia infraestructura determina en qué recurso de cómputo concreto debe ejecutarse cada tarea. Por otro lado, el sub-flujo de trabajo *Get_job_results* monitoriza la ejecución de las tareas enviadas a la infraestructura y recupera los resultados producidos. También analiza si las tareas han sido ejecutadas correctamente y si los datos son los esperados, proporcionando los datos de salida de la aplicación.

3.4. Resultados de la experimentación

En la figura 3.5 se representa el número de términos que fueron asignados a cada infraestructura de computación, así como el total de términos asignados a la infraestructura en su conjunto con el fin de procesar los 15 millones de objetos, generando cerca de 150 millones de términos. La anotación de los 149.427.907 términos fue realizada en un total de 4.260,83 horas (177,43 días), obteniendo un gran avance respecto de la estimación inicial de 1.640 años de CPU que se necesitarían utilizando un único procesador.

Para lograr esta mejora fue necesario aprovechar al máximo la capacidad de computación a nuestra disposición. Las infraestructuras HERMES, AraGrid y PireGrid no estaban dedicadas a la resolución de este problema, sino que eran compartidas con el resto de grupos de

investigación que hacen uso diario de ellas. Además, nuestro perfil de usuario tenía prioridad normal, lo que limitó el uso de recursos siempre por debajo del 25 % de la capacidad de los mismos.

Asimismo, la paralelización de ADEGA original contribuyó a la reducción del coste computacional del algoritmo. Experimentalmente se había determinado el tamaño de tarea ejecutada en cada infraestructura: en el caso de HERMES, cada tarea anotaba 100 términos, y en el caso de AraGrid y PireGrid, 400 términos por tarea. En base a este tamaño, para completar el proceso de anotación se asignaron 352.745 tareas a HERMES (35.274.419 términos), 186.210 a AraGrid (correspondientes a 74.4833.721 términos) y finalmente 99.175 tareas fueron asignadas a PireGrid (con un total de 39.669.767 términos). Desde la perspectiva de paralelización de tareas, HERMES tuvo una media de 164 trabajos procesados en paralelo, 116 para AraGrid y 68 para PireGrid. Por otro lado, los resultados obtenidos también demuestran que se obtienen mejores prestaciones si se utiliza la infraestructura de integración para ejecutar la aplicación en vez de una infraestructura de computación concreta. En este sentido, la mejora del rendimiento obtenida desde la perspectiva del usuario final es de 2,0 con respecto a AraGrid, 3,6 con respecto a PireGrid, y 3,8 con respecto a HERMES.

Otra medida de interés es el tiempo medio invertido en anotar un término en cada infraestructura, desde el punto de vista del sistema (tiempo efectivo de ejecución, ignorando los distintos tipo de retardos producidos por la gestión interna de las infraestructuras) y desde el punto de vista del usuario final (la aplicación ejecutada). En el caso de HERMES, los tiempos medios obtenidos fueron de 71,18 segundos y 222,8 segundos, respectivamente. En el caso de AraGrid, los tiempos mejoraron considerablemente: 23,84 segundos y 41,56 segundos, respectivamente. Estos resultados son muy similares a los obtenidos en PireGrid, 26,09 segundos y 43,70 segundos.

Por último, durante la ejecución de la aplicación también se midió el porcentaje de tareas que experimentaron un fallo: concretamente, la tasa de fallo se situó en el 12,39 % de las tareas inicialmente ejecutadas. Dada la naturaleza de este tipo de infraestructuras, era previsible que se produjeran fallos, motivo por el cual el sistema dispone de mecanismos de tolerancia a fallos, que permiten gestionar los fallos de forma totalmente transparente para el usuario final. En este trabajo concreto, el mecanismo de tolerancia a fallos empleado consiste en aplicar una sencilla política de re-planificación: en primer lugar, si una tarea falla, se vuelve a enviar a la misma infraestructura donde inicialmente se había pretendido ejecutar; si vuelve a fallar, se envía para su ejecución a otra infraestructura de computación diferente (para este propósito

se utilizó AraGrid, ya que se comprobó experimentalmente que era la infraestructura con una menor tasa de fallos); y si el fallo persiste, se ejecuta utilizando recursos de computación de Amazon EC2. Aplicando esta política, el 56,25 % de las tareas que fallaron inicialmente fueron correctamente ejecutadas aplicando un reenvío sobre la misma infraestructura; el 41,24 % restante fueron resueltas utilizando una infraestructura alternativa; y sólo el 2,5 % del total de tareas que experimentaron un fallo tuvieron que ser ejecutadas en instancias *m1.medium* de Amazon EC2. Al final, todas las tareas fueron ejecutadas con éxito.

3.5. Conclusiones

En este capítulo hemos resuelto el problema de la generación de datos semánticos a gran escala derivado del uso de la versión original de ADEGA, concretamente la anotación semántica de los objetos de aprendizaje de Universia. Como parte del mismo se ha utilizado una infraestructura de recursos computacionales heterogéneos y tecnologías semánticas para implementar una aplicación responsable de generar los grafos de la DBpedia que anoten los objetos de Universia. Internamente, la infraestructura maneja la heterogeneidad de los diferentes modelos computacionales utilizados y el ciclo de vida de las tareas, para anotar semánticamente los objetos. Una cuestión importante en esta aproximación era determinar la configuración más apropiada para el despliegue de la aplicación programada. Hemos analizado el comportamiento computacional de cada recurso disponible para encontrar su mejor configuración, llevando a cabo este análisis en el ámbito de una nueva metodología capaz de configurar experimentalmente las aplicaciones que serían ejecutadas en la infraestructura. La flexibilidad que ofrece la infraestructura desde el punto de vista del programador y su capacidad para integrar diversos recursos supone una diferencia fundamental e importante respecto a otras experiencias similares.

Si bien existe una enorme mejora a la hora de anotar objetos de aprendizaje, para cada objeto se tarda una media de 715 segundos, lo cual sigue siendo un tiempo prohibitivo para una variedad de aplicaciones en las que se requiere anotar grandes cantidades de información en poco tiempo (como, por ejemplo, el texto generado por usuarios en redes sociales como Facebook o Twitter, las wikis colaborativas, o los periódicos online). Por esta razón se requiere no sólo un cambio del algoritmo de exploración, ya que una solución DFS consume un gran tiempo de acceso a los datos (debido a la complejidad y profundidad de la DBpedia), sino también un cambio en la tecnología usada para acceder a las características de los nodos

y para procesar el texto de cada uno de ellos. En este sentido, el uso de Virtuoso y otras tecnologías RDF para almacenar y consultar los repositorios como la DBpedia pueden ser un cuello de botella en términos de rendimiento. Otras soluciones de almacenamiento basadas en la computación en la nube, o las bases de datos NoSQL por ejemplo, podrían ser evaluadas como una alternativa.

Capítulo 4

Mejora del rendimiento en ADEGA

4.1. Introducción

En el capítulo 3 se enunciaban los principales problemas que tenía ADEGA a partir de un estudio estadístico de los indicadores de la versión secuencial del algoritmo (ver tabla 3.1) y se indicaba la necesidad de paralelizarlo para abordar los problemas de rendimiento a la hora de anotar documentos; más concretamente, de explorar y filtrar la DBpedia. Ejecutar el algoritmo en un entorno distribuido disminuye el tiempo de anotación de un término, ya que el algoritmo se paraleliza en varias máquinas y cada una se dedica a explorar una rama del grafo final reduciendo el tiempo total del algoritmo. Ahora bien, esta disminución, aun siendo significativa, no es suficiente para algunas aplicaciones, ya que el tiempo de exploración de un nodo raíz consume de media 71 segundos. Por otra parte, es difícil tener siempre a nuestra disposición un entorno distribuido del tamaño del utilizado para resolver el problema de anotación de Universia. Incluso en ese caso, se dispuso de un perfil de usuario de prioridad normal donde se limitaba el uso de los recursos computacionales por debajo del 25 % de la capacidad de los mismos. Además, hay que tener en cuenta el coste asociado al uso de las infraestructuras computacionales, que en el caso del repositorio de Universia consistió en unos 40.000 € para resolver el problema de rendimiento.

Son varios los factores que contribuyen al bajo rendimiento de ADEGA. El principal es el coste de exploración del grafo, que se incrementa a medida que se aumenta la profundidad de exploración (siendo $O(b^n)$, donde b es el factor de ramificación y n es el nivel más profundo de exploración). Este coste repercute directamente en el tiempo de ejecución del algoritmo, concretamente en la exploración y filtrado, impidiendo realizar exploraciones a niveles altos

de profundidad para valorar la mejora en precisión y cobertura.

Otro cuello de botella importante es la tecnología utilizada para obtener las relaciones de la ontología. Los indicadores obtenidos para la versión secuencial de ADEGA reflejan que el 67 % del tiempo de cómputo del algoritmo se dedica al acceso y consulta de la DBpedia. En este caso, los datos relativos a la ontología están almacenados en una base de datos semántica, más concretamente *OpenLink Virtuoso*[1]. Este tipo de bases de datos permiten representar las relaciones semánticas entre las diferentes entidades y formular consultas mucho más expresivas, donde se realiza cierto grado de inferencia para obtener nuevos recursos que serían imposibles con otro tipo de sistemas. A pesar de estas ventajas, su rendimiento es muy deficiente: los tiempos de acceso y recuperación de los datos son muy lentos, incrementando en definitiva el tiempo de ejecución del algoritmo. Teniendo en cuenta que se realizan consultas simples para obtener los nodos hijos del nodo en exploración, estos beneficios a la hora de realizar la exploración del grafo no suponen ninguna ventaja. Por lo tanto, considerar otras soluciones de almacenamiento que permitan mejorar estos tiempos, como las bases de datos NoSQL (no relacionales), parece una alternativa viable.

Del mismo análisis de indicadores, podemos extraer que de la cantidad de nodos procesados por el algoritmo (48.573,29 de media por cada término), el 91 % de ellos son nodos de datos primitivos; es decir, texto que necesita ser analizado en tiempo de ejecución del algoritmo mediante técnicas de procesamiento de lenguaje natural a medida que se van obteniendo los nodos. En la fórmula 2.5 utilizada para calcular la relevancia de los datos primitivos en función del contexto, se puede ver que ciertas partes de la fórmula, concremante la frecuencia termonológica, puede venir precalculada si el contenido textual de la ontología es pre-procesado con anterioridad. De esta forma, evitaríamos cálculos innecesarios durante el tiempo de ejecución del algoritmo.

Para dar solución a estos problemas, en este capítulo proponemos una nueva versión de ADEGA que mejore el rendimiento y permita disminuir el tiempo de exploración del grafo de la DBpedia, reduciendo el tiempo final de creación de la solución de anotación sin necesidad de contar con un entorno distribuido para la ejecución del algoritmo. En las secciones siguientes se explica la importancia de un pre-procesamiento de la DBpedia, el cambio de las tecnologías utilizadas y las modificaciones llevadas a cabo tanto en el contexto como en el algoritmo de filtrado.

[1] https://virtuoso.openlinksw.com/

Relación	Número de instancias
dbo:wikiPageWikilink	48.704.874
dcterms:subject	13.610.094
rdfs:type	11.168.302
dbo:wikiPageRedirects	5.074.113
skos:broader	1.463.237
dbo:wikiPageDisambiguates	1.004.742
dbo:birthPlace	979.163
dboprop:subdivisionType	598.602
dbo:country	557.587
Más de 15K de relaciones diferentes	20.304.358

Tabla 4.1: Cómputo del número de instancias de las relaciones en la DBpedia. Datos correspondientes a la versión 3.7 de la DBpedia.

4.2. Pre-procesamiento de la ontología

Para evitar el cuello de botella producido al acceder y consultar la DBpedia en Virtuoso, las relaciones entre conceptos se almacenaron en bases de datos NoSQL[2,3]. Estos sistemas difieren de los clásicos relacionales en aspectos importantes: no usan el mismo lenguaje de consulta SQL; los datos almacenados no requieren estructuras fijas como tablas, sino que funcionan con estructuras clave-valor u orientadas a documentos; y en general no soportan operaciones de agregación ni garantizan la transaccionalidad (es decir, las características ACID de atomicidad, consistencia, aislamiento y durabilidad). Una de sus grandes ventajas es que escalan bien horizontalmente, pueden manejar grandes cantidades de datos sin generar cuellos de botella y están diseñadas para que se ejecuten en clústeres de máquinas de bajas prestaciones para distribuir la base de datos y no centralizar el acceso.

Es necesario mencionar que nuestra aproximación no indexa todas las relaciones del repositorio en la base de datos NoSQL, sino únicamente aquellas *relaciones entre entidades* que nos permiten construir la topología del grafo. En la tabla 4.1 se han analizado las relaciones más utilizadas del repositorio, un total de 103.465.072 instancias. Hay más de 15.000 relaciones diferentes, pero no todas tienen la misma importancia, ya que hay un pequeño grupo de relaciones que unen a la mayor parte de los nodos del grafo; por lo tanto, la necesidad de agregar todas las relaciones finalmente se reduce a un cierto porcentaje significativo. Como se puede

[2]https://www.mongodb.com/es
[3]http://cassandra.apache.org/

ver en la tabla, la relación más numerosa, dbo:wikiPageWikiLink, supone el 47 % del total de relaciones; estas relaciones se obtienen a partir de los enlaces internos entre artículos de la Wikipedia. Las relaciones dcterms:subject con una representación del 13 % del total y skos:broader con 1,4 %, son relaciones que expresan jerarquía entre entidades; mientras que rdfs:type con un 10,7 % representa la pertenencia a una clase concreta en la ontología. Dentro del conjunto de relaciones, dbo:wikiPageRedirects y dbo:wiki-PageDisambiguates no son útiles para la construcción del grafo, sino que aportan otro tipo de información muy valiosa. En el caso de las relaciones dbo:wikiPageRedirects, enlazan a entidades con nombres alternativos a la entidad principal, correspondiéndose con sus sinónimos; mientras que dbo:wikiPageDisambiguates proporciona enlaces a otras entidades que presentan conflictos por proceder de un mismo término ambiguo.

Teniendo esto en cuenta, podemos seguir una aproximación incremental a la hora de almacenar los datos de la DBpedia, maximizando el número de relaciones útiles para la construcción del grafo, y por lo tanto, no penalizando la exploración. Analizando nuestros grafos solución, prácticamente la totalidad de las entidades son accedidas a través de las relaciones jerárquicas; mientras que muy pocas entidades son accedidas usando la relación dbo:wikiPageWikilink a pesar del gran número de instancias, ya que aunque proporcione enlaces internos dentro del artículo, estos no tienen por qué estar necesariamente relacionados con la entidad. Por otro lado, la relación rdfs:type suele sustituirse con dcterms:subject, ya que proporciona una clasificación mucho más específica. Por estos motivos, las relaciones que serán almacenadas en la base de datos NoSQL son inicialmente las jerárquicas y el resto de relaciones específicas para cada entidad, cada una de las cuales tiene un porcentaje de representación bajo, pero su valor añadido es alto debido a la adecuación de la información a la que enlaza. Teniendo en cuenta lo mencionado anteriormente, se descartaron las relaciones dbo:wikiPageWikilink, rdfs:type, dbo:wikiPageDisambiguates y dbo:wikiPageRedirects.

Las *relaciones hacia datos primitivos* requieren de procesamiento antes de ser indexadas. Para mejorar el acceso a los datos textuales y dinamizar el cálculo de la similitud entre las relaciones textuales y el contexto, hemos utilizado Lucene[4], ya que nos permite crear índices, utilizar su sistema para calcular similitudes entre dos documentos y explotar su motor de búsqueda para obtener resultados similares a los datos de la consulta. Para favorecer la

[4]Lucene es una librería de código abierto para indexar y realizar búsquedas a texto completo. Para más información consultar http://lucene.apache.org/

Campo	Valor de ejemplo
Palabra clave	*Coach*
Término nominal	*Coach*
Sinónimos	*Coaches, Coach seat*
Desambiguación	*http://dbpedia.org/resource/Coach_(sport)*
	http://dbpedia.org/resource/Coach_(TV_series)
	http://dbpedia.org/resource/Jonathan_Coachman
URI de la entidad	*http://dbpedia.org/resource/Coach*

Tabla 4.2: Ejemplo de documento del *índice de etiquetas* de la DBpedia para la entidad *Coach*

flexibilidad y la independencia del tipo de fichero indexado, la estructura interna de Lucene almacena los ficheros como *Documentos* que contienen *Campos*. En nuestro caso, hemos considerado la creación de *dos* índices sobre los datos:

- **Índice de etiquetas**: se utiliza para mejorar la detección de términos en el texto, y en él se indexan las etiquetas de las entidades. En la tabla 4.2 se listan los 5 campos que conforman el índice. El campo *palabra clave* es la etiqueta de la entidad obtenida de la relación `rdfs:label`, el *término nominal* es el término que representa al conjunto de variaciones del mismo, los *sinónimos* son obtenidos a través de la relación `dbo:wikiPageRedirects`; las URIs de *desambiguación*, se obtienen a través de `dbo:wikiPageDisambiguates`; y finalmente, el campo *URI de la entidad* contiene la URL del recurso en la DBpedia.
- **Índice textual**: contiene todo el texto procesable de la DBpedia para cada una de las entidades. Su estructura se puede ver en la tabla 4.3 y prácticamente es la misma que la del índice de etiquetas, sin los enlaces de desambiguación y con las descripciones de cada una de las entidades, obtenidas de la relación `dbo:abstract`. A mayores, contiene un campo de agregación, *texto completo*, que contiene todo el texto disponible para la entidad.

4.3. Extracción del contexto mediante *n-gramas*

En la creación del contexto, hemos sustituido la extracción de términos en tres fases (análisis de morfología, similitud y frecuencia) por un nuevo algoritmo que utiliza como base un

Campo	Valor de ejemplo
Palabra clave	*Coach (sport)*
Término nominal	*Coach (sport)*
Sinónimos	*Sports coaching, Assistant coach, Sports instructor, ...*
Resumen	*In sports, a coach is a person involved in the direction, instruction and training of the operations of a sports team or of individual sportspeople. A coach may also be a teacher.*
Texto completo	Concatenación de todos los campos texuales.
URI de la entidad	*http://dbpedia.org/resource/Coach_(sport)*

Tabla 4.3: Ejemplo de documento del *índice textual* de la DBpedia para la entidad *Coach_(sport)*

detector de *n-gramas*, mejorando con ello el reconocimiento de términos simples y compuestos anteriormente identificados mediante reglas JAPE en el análisis morfológico. Este tipo de algoritmos se usan con frecuencia en procesamiento de lenguaje natural y tienen como objetivo seleccionar una subsecuencia de *n* palabras en el texto de entrada.

4.3.1. Detección de *n-gramas* y morfología

En la versión anterior del algoritmo, se definían a priori y de forma manual un conjunto de reglas heurísticas JAPE, descritas en la figura 2.3, para identificar los términos compuestos del texto. En el caso de obtener una detección incorrecta de los mismos, había que introducir una nueva regla que corrigiera el problema y detectase el término. Además, en el caso de cambiar de idioma, las reglas necesitaban reformularse debido a la dependencia del lenguaje en la construcción sintáctica y morfológica de las frases.

La finalidad del algoritmo de *n-gramas* es reconocer correctamente términos simples y compuestos en un documento de texto a partir de un índice de términos posibles, sin recurrir a reglas heurísticas; de forma que esta tarea se realice automáticamente y sea independiente del lenguaje. En nuestro caso concreto, hemos configurado el algoritmo para la detección de pentagramas, palabras compuestas por cinco términos como máximo, ya que en nuestra experimentación con la versión anterior de ADEGA no se recuperaron términos compuestos de mayor longitud. El índice de todos los posibles términos a detectar es el creado a partir de las etiquetas de la DBpedia, cuya estructura puede consultarse en la tabla 4.2. Mediante el buscador proporcionado por Lucene es fácil realizar una consulta con el término candidato extraído del texto y comprobar si hay entidades en la DBpedia que tengan como etiqueta la misma

Algoritmo 4.1: Algoritmo de *n-gramas*

Input: Texto a analizar d
Output: Lista de términos detectados $lista_t$
Data: Ventana activa N

$ventana \leftarrow N$;
$lista_{morf} \leftarrow$ detectar términos con morfología permitida en d;
$lista_f \leftarrow$ detectar frases en d;
for $frase \in lista_f$ **do**
  $lista_t \leftarrow$ detectar términos en $frase$;
  for $i \leftarrow 0$ **to** $lista_t$ **do**
    /* Obtener ventana activa para el término $i-ésimo$ */
    if $i + N >$ *longitud de* $lista_t$ **then**
      $ventana \leftarrow$ i + longitud de $lista_t$;
    end
    for $j \leftarrow ventana$ **to** *0* **do**
      $cadena_t \leftarrow$ obtener términos de $lista_t$ desde i hasta $i+j$;
      $lista_r \leftarrow$ buscar $cadena_t$ en índice de etiquetas;
      if $lista_r \neq \emptyset$ **then**
        if *longitud de* $cadena_t$ *= 1* **and** $cadena_t \in lista_{morf}$ **then**
          Añadir $cadena_t$ a $lista_t$;
          if *no permitir sopalamiento* **then**
            $i \leftarrow i+j-1$;
            break;
          end
        else if *longitud de* $cadena_t > 1$ **then**
          Añadir $cadena_t$ a $lista_t$;
          if *no permitir sopalamiento* **then**
            $i \leftarrow i+j-1$;
            break;
          end
        end
      else if $lista_r = \emptyset$ **and** *longitud de* $cadena_t$ *= 1* **and** $cadena_t \in lista_{morf}$ **then**
        Añadir $cadena_t$ a $lista_t$;
        if *no permitir sopalamiento* **then**
          $i \leftarrow i+j-1$;
          break;
        end
      end
    end
  end
  $ventana \leftarrow N$;
end

palabra clave que la de la consulta. Utilizando esta premisa, el algoritmo 4.1 selecciona una *ventana activa* de máximo N términos y realiza consultas al índice para comprobar si hay entidades que coincidan con el término de la búsqueda (línea 11). Si no hay coincidencia, vuelve a realizar la consulta para $N-1$ términos de la ventana hasta encontrar una coincidencia con una etiqueta de una entidad existente en el índice, que en ese caso avanza la ventana activa hasta el siguiente conjunto al término detectado (líneas 16,22 y 29); o hasta agotar todos los términos, en cuyo caso desplaza una posición a la derecha la ventana y vuelve a comenzar realizando la consulta con N términos nuevamente (línea 36). En el caso de $N = 1$ se comprueba la morfología del término, que al igual que se hacía antes, se descarta si su morfología es verbo, conjunción, preposición o determinante (línea 14 y 27).

Por ejemplo, si partimos del texto *Amanecer en Meoto Iwa o "Rocas Casadas", en la ciudad de Ise, Japón.* y definimos el N inicial como $N = 4$, la primera ventana activa sería *Amanecer en Meoto Iwa*. Se comprueba mediante una consulta si existe una entidad con esa etiqueta. Dado que no existe, se vuelve a realizar una consulta para $N-1$ términos: *Amanecer en Meoto*. Se siguen realizando consultas con menos términos hasta obtener un resultado o conseguir $N = 1$. En este caso, con $N = 1$, tendríamos *Amanecer*, que coincide con una entidad de la DBpedia y tiene una morfología permitida, por lo tanto se acepta como término del contexto. Se mueve la ventana activa un término hacia la derecha, por lo tanto lo siguiente a procesar sería *en Meoto Iwa o*. Se realiza el mismo procedimiento que hemos descrito, hasta encontrar una posible entidad al realizar las consultas o acabar la ventana. Dado que las combinaciones de términos en esa ventana no generan ninguna posible etiqueta, se desplaza la ventana activa un término a la derecha para procesar *Meoto Iwa o Rocas*. En esta ventana, con N=2 obtendríamos una entidad que coincide con la etiqueta *Meoto Iwa*. En este ejemplo propuesto, los términos aceptados para el contexto serían *amanecer, Meoto Iwa, Rocas Casadas, ciudad, Ise, Japón*.

Para detectar la morfología de las palabras, hemos sustituido GATE por el etiquetador de morfología de la Universidad de Stanford: *Stanford Log-linear Part-Of-Speech Tagger* [102][5], debido a que su clasificador basado en máxima entropía alcanza una precisión del 98 %.

A través de este este algoritmo, detectamos los términos del documento que tienen alguna entrada en la DBpedia, en vez de obtener directamente todos los términos simples y compuestos que son nombres o adjetivos. De este modo, entendemos que se mejora la obtención del contexto al estar formado por términos obtenidos del propio repositorio con el que se realiza

[5] http://nlp.stanford.edu/software/tagger.shtml

la anotación.

4.3.2. Frecuencias, agrupación de términos similares y relevancia

Debido a que en el índice de etiquetas se ha introducido un campo de sinónimos, podemos identificar los términos presentes en el texto que son sinónimos de un término t y agruparlos para sumar sus frecuencias. El término nominal será el término elegido para representar al conjunto. De esta forma, la frecuencia se calcula como sigue:

$$tf_{t,\alpha} = (num_{t,\alpha} + sin_{t,\alpha})^{\frac{1}{2}} \tag{4.1}$$

donde:

- α es un campo del documento estructurado a procesar. Si no está estructurado, se asume que solo hay un campo.
- $num_{t,\alpha}$ es el número de ocurrencias de t en α.
- $sin_{t,\alpha}$ es el número de ocurrencias de los sinónimos de t identificados en el texto.

Finalmente calculamos la relevancia de un término para ordenarlo en función de su importancia dentro del documento siguiendo la fórmula de la ecuación 2.2.

4.4. Filtrado del grafo basado en contexto

El objetivo de modificar el algoritmo de filtrado es reducir la complejidad de la exploración, de modo que se mejora su rendimiento, entendido como el tiempo necesario para obtener una solución. En este sentido, la necesidad de filtrar la información es primordial, pero puede acotarse de forma que no sea necesario realizar una exploración exhaustiva del grafo de la DBpedia con cada anotación. En la descripción del algoritmo de filtrado realizada en la sección 2.3, explicábamos que utilizábamos un algoritmo DFS limitado en profundidad para explorar el grafo partiendo de un conjunto de nodos raíz, correspondientes a las entidades que representan a los términos del contexto. El problema de esta aproximación es que se necesita explorar de forma exhaustiva *todos* los nodos, sin usar ningún tipo de conocimiento o heurística para podar ciertas ramas del grafo y evitar su procesamiento. Esto nos llevaba a explorar para cada nodo raíz una media de más de 48 millones de nodos, para una exploración

de profundidad tres (tabla 3.1). Sin embargo, si tenemos en cuenta que estos datos fueron obtenidos para contextos con una media de diez términos, no tiene sentido que se creen grafos tan extensos, porque la mayor parte de su contenido no va a estar relacionado.

Para paliar la sobre-exploración del grafo, donde la mayor parte de las entidades son descartadas, hemos convertido el problema de exploración a un problema de búsqueda de caminos en un grafo con la incorporación de un conjunto de *nodos finales*. Estos nodos finales son un conjunto de entidades con una gran similitud con los términos del contexto. Tiene sentido pensar que los nodos que se encuentren en el camino entre la entidad raíz que describe al término del contexto y las entidades del conjunto de nodos finales estarán bastante relacionados con el contexto, aportando mayor conocimiento y complementando la información. De esta forma, evitamos tener que procesar una mayor cantidad de nodos que, con gran probabilidad, no tendrán una temática relacionada.

Para la obtención de los *nodos finales*, utilizamos la información textual de la DBpedia almacenada en el índice de la tabla 4.3, con el fin de buscar nodos similares a los términos del contexto. Mediante una consulta booleana en Lucene utilizando todos los términos del contexto enlazados con un operador *OR*, recuperamos los nodos que comparten la mayor cantidad de términos con los de la consulta. Para mejorar las posibilidades de encontrar los nodos que tengan más términos del contexto, incluimos en la búsqueda los posibles sinónimos de cada uno, de forma que puedan encontrarse nodos que usen los términos en su forma nominal o en sus variaciones. Además, para potenciar la importancia de cada término dentro de la búsqueda, se asigna un peso a cada uno correspondiente a su relevancia dentro del contexto. De esta forma, los nodos que tengan términos con mayor relevancia serán más importantes en el ranking de la búsqueda que aquellos que compartan términos de poca relevancia.

En cuanto a la obtención de los *nodos raíz*, la búsqueda se realiza en el índice de etiquetas de la DBpedia utilizando un único término del contexto para cada nodo mediante una consulta con el término nominal. Aplicamos un conjunto de heurísticas sobre los resultados basadas en la calificación obtenida en la consulta, la construcción del término y su sintáctica para seleccionar el resultado correcto. En el caso de que el resultado sea un nodo de desambiguación, necesitamos saber cuál de todas las posibles acepciones es la correcta. Para ello, aplicamos sobre ese nodo nuestro propio algoritmo de filtrado con baja profundidad de exploración para seleccionar la acepción correcta. Finalmente, devolvemos aquel nodo que obtenga mayor relevancia.

Algoritmo 4.2: Algoritmo de exploración y filtrado DFS bidireccional

Input: Nodo inicial n_s, nodo final n_f
Output: Grafo solución λ
Data: Profundidad de exploración d, S_s y S_f conjuntos de nodos de los subgrafos λ_s y λ_f del nodo inicial y final respectivamente; S_i, conjunto de nodos de intersección

1 $\lambda_s \leftarrow$ `iterativeDFS(`n_s,$d/2$`)`;
2 $\lambda_f \leftarrow$ `iterativeDFS(`n_f,$d/2$`)`;
3 $S_i \leftarrow$ `interseccion(`S_s *de* λ_s, S_f *de* λ_f`)`;
4 Añadir n_s y n_f a λ;
5 **for** $n \in S_i$ **do**
6 | Añadir n a λ;
7 | `construirCamino(`n,λ_s`)` y añadir a λ;
8 | `construirCamino(`n,λ_f`)` y añadir a λ ;
9 **end**

4.4.1. Algoritmo de búsqueda de caminos

El objetivo de este algoritmo es descubrir los caminos entre el conjunto de *nodos raíz* y el conjunto de *nodos finales*. Para cada nodo raíz y final, se aplica una estrategia bidireccional (ver algoritmo 4.2), que consiste en dos búsquedas DFS que parten desde cada nodo, intentando encontrar un camino entre ambos (líneas 1 y 2 del algoritmo). La profundidad de exploración en cada búsqueda se divide entre las dos exploraciones, de tal forma que las dos búsquedas se encuentran a la mitad del camino. La obtención de las relaciones de primer nivel para un nodo se realiza mediante una consulta a la base de datos NoSQL. Para crear el grafo final, se intersecan los conjuntos de nodos de los dos subgrafos (línea 3) y se unen los caminos parciales cuyos nodos frontera coincidan (líneas 7 y 8). El resto de nodos que no han generado caminos se descartan. Hay que puntualizar que no solo interesa el camino más corto cuando buscamos caminos entre los nodos, sino todos los caminos posibles de una profundidad máxima establecida. Hay ciertas modificaciones llevadas a cabo en el algoritmo DFS en comparación a la implementación descrita en la sección 2.3:

- La exploración es genérica, independiente de las relaciones del grafo, para que pueda ser aplicada a cualquier ontología, no solamente a la estructura de relaciones de la DBpedia.
- El pesado de los nodos (explicado en la subsección 4.4.2) es independiente de la exploración y se realiza a posteriori.

- Se procesan todas las relaciones sin establecer un umbral de cardinalidad, ya que aunque podría caerse en una relación multievaluada, pueden perderse caminos entre los nodos si no se exploran todas las instancias.

- La implementación del algoritmo es iterativa, eliminando la recursividad.

4.4.2. Evaluación de nodos

A diferencia de la estrategia de evaluación de nodos que seguíamos en la sección 2.3.2, en la que considerábamos dos evaluaciones diferentes en función del tipo de nodo, entidades o datos primitivos; en la nueva versión del algoritmo, esta diferenciación no se aplica, ya que los datos primitivos son intrínsecos a las entidades que representan. En este sentido, toda entidad tiene una etiqueta con la que se le asigna un nombre y la mayor parte de las entidades tienen asociada una descripción que explica el concepto al que hacen referencia. En el cálculo de relevancia de un nodo *n* entendemos que hay una contribución *(i)* en el grafo mediante las relaciones entre éste y otros nodos; y *(ii)* una contribución de la propia entidad con respecto al contexto, que viene dada por los datos primitivos. De este modo, ya no existe una diferenciación entre nodos de datos primitivos y nodos entidades, sino que todos los nodos son entidades que representan a un concepto y tienen asociado un conjunto de datos primitivos que los describen.

Contribución de un nodo según su localización en el grafo

Para medir la importancia relativa que tiene un nodo dentro de la topología del grafo, hemos utilizado una *medida de centralidad* para saber cómo influyen las relaciones entre los nodos. Esta medida es un valor estructural que depende solamente de su localización dentro del grafo. En este sentido, sólo se evalúa la contribución de un nodo según su ubicación.

Existen diferentes medidas de centralidad que se pueden usar para pesar un grafo. La medida de *centralidad de grado* se corresponde con el número de enlaces que posee un nodo con los demás. En el caso de los grafos dirigidos, se pueden definir dos medidas de centralidad de grado diferentes, correspondientes al grado de entrada y al de salida. En el caso de nodos que representan a conceptos, podemos entender el grado de entrada como la popularidad de un concepto frente a otros; mientras que el grado de salida representaría la integración del concepto en otros. Existen otras medidas más complejas basadas en la centralidad de grado, pero añadiendo información de importancia relativa dentro de un grafo: la centralidad del

vector propio. La idea es que los nodos que poseen un valor alto de esta medida de centralidad están conectados a muchos nodos que a su vez están bien conectados. Los nodos más centrales corresponden a centros de grandes grupos cohesivos, donde la conexión entre los nodos pesa de forma diferente; mientras que en el caso de la centralidad de grado, cada nodo pesa lo mismo dentro de la red. Los algoritmos *PageRank* [84] y *HITS* [54] implementan la idea de centralidad del vector propio.

En el estudio de [76], en el que se comparan diferentes medidas de conectividad en grafos aplicadas al problema de desambiguación lingüística de términos, se concluye que las mejores medidas para resolver este tipo de problemas son la medida de centralidad de grado y PageRank. Tendiendo en cuenta que la centralidad de grado tiene una complejidad de $O(n)$, inferior comparada con la del PageRank $O(n^2)$, y que los resultados son semejantes, hemos escogido la medida de centralidad de grado para evaluar la importancia de un nodo dentro de un grafo solución.

Formalmente, dado un grafo $G := (N,R)$, donde N es el conjunto de nodos y R su conjunto de relaciones, para cada nodo $n \in N$, su centralidad de grado $C_{DEG}(n)$ se define como:

$$C_{DEG}(n) = grado_{entrada}(n) + grado_{salida}(n) \tag{4.2}$$

Correspondiéndose $grado_{entrada}(n)$ y $grado_{salida}(n)$ con el número de relaciones entrantes y salientes del nodo n, respectivamente.

Contribución de un nodo con respecto al contexto

Para poder determinar la relevancia o influencia que tiene un nodo en relación al texto de partida, utilizaremos la similitud entre los datos primitivos de un nodo y el contexto generado inicialmente.

Para calcular la similitud entre el contexto Δ y los datos primitivos de una entidad n_t, $sim(\Delta, n_t)$, utilizamos la implementación de similitud de Lucene basada en el modelo de espacio vectorial [92]. Es un modelo algebraico utilizado para filtrar, recuperar, indexar y calcular relevancias de información, representando documentos escritos en lenguaje natural de una manera formal mediante el uso de vectores en un espacio lineal multidimensional. Se basa en la idea de que la relevancia de un documento frente a una búsqueda puede calcularse usando la diferencia de ángulos entre los vectores que representan a los términos de la búsqueda y a los del documento, calculándolo como el coseno del ángulo entre sendos vectores (similitud del coseno). Un valor de coseno de cero significa que la búsqueda y el documento son orto-

gonales el uno al otro, por lo tanto, no hay coincidencia. Para calcular el coseno del ángulo entre dos vectores se usa la siguiente ecuación:

$$sim_{cos}(q,d) = \frac{V_q \cdot V_d}{|V_q||V_d|} \tag{4.3}$$

donde:

- V_q y V_d son los vectores de la consulta y el documento, respectivamente.
- $V_q \cdot V_d$ es el producto escalar de los vectores
- $|V_q|$ y $|V_d|$ son las distancias euclídeas.

La ecuación anterior se puede ver como el producto escalar de los vectores normalizados, en el sentido de que Vq se normaliza a un vector unitario dividiendo por su distancia euclídea. Lucene refina la fórmula anterior teniendo en cuenta que:

- Puede ser problemático normalizar V_d a su vector unitario, porque se elimina toda la información de longitud del documento. Para algunos documentos esto es correcto, porque puede darse el caso de que el documento esté compuesto por párrafos duplicados, pero para documentos cuya información no está duplicada es una mala consideración. Para evitar este problema, se usa una un factor de normalización diferente para la longitud del documento, donde se normaliza a un vector igual o mayor al vector unitario, $norm_d$.
- En tiempo de indexación se puede especificar que ciertos documentos son más importantes que otros asignando un *peso*. Para tener esto en cuenta, cada documento se multiplica por su peso, p_d.
- Durante la búsqueda, se pueden establecer pesos para la consulta, subconsultas o los propios términos de la consulta; por lo tanto, la contribución de una consulta a la calificación del documento en el ranking se multiplica por el peso, p_q.
- Un documento puede coincidir con una consulta de varios términos sin necesidad de contenerlos a todos, en este sentido los usuarios pueden pesar a mayores los documentos que contengan más cantidad de términos a través de un factor de coordinación, que suele asignar un peso mayor cuantos más términos contengan los documentos, $coord_{q,d}$.

Bajo una suposición simplista de que un documento indexado está formado por un único campo, Lucene propone la siguiente fórmula conceptual:

$$sim(q,d) = coord_{q,d} \cdot p_q \cdot \frac{V_q \cdot V_d}{|V_q|} \cdot norm_d \cdot p_d \tag{4.4}$$

Para un cálculo eficiente de la similitud, algunos componentes de la fórmula se calculan con antelación:

- $coord_{q,d}$ es el factor de coordinación, que asigna más peso cuantos más términos contengan los documentos.
- p_q es el peso asignado a un término, establecido en la consulta.
- $|V_q|$ distancia euclídea del vector de la consulta. Se puede calcular cuando comienza la búsqueda, porque es independiente de los documentos a evaluar. Desde una perspectiva de optimización de la búsqueda, puede preguntarse por qué es necesario normalizar la consulta, ya que todos los documentos se multiplicarán por este valor ($|V_q|$) haciendo que el ranking no se vea afectado por esta normalización. Hay dos razones para realizarla: la primera, es que los valores de similitud para el mismo documento con dos consultas distintas tienen que ser comparables; y la segunda, es que ayuda a que la similitud calculada ronde el vector unitario para prevenir la pérdida de datos debida a la precisión del punto flotante.
- $norm_d$ y p_d, factor de normalización y peso del documento, respectivamente, se conocen en tiempo de indexación del documento, por lo que se calcula su multiplicación y se almacena como un único valor, n_d.

La ecuación final para calcular la similitud entre una consulta (que en nuestro caso correspondería a todos los términos del contexto) y un documento indexado (los datos primitivos de las entidades de la DBpedia) es la siguiente:

$$sim(q,d) = coord_{q,d} \cdot n_q \cdot \sum_{t \in q} \left(tf_{t,d} \cdot idf_t^2 \cdot p_t \cdot n_{t,d} \right) \tag{4.5}$$

donde:

- $tf_{t,d}$ es la frecuencia de un término, que se corresponde con el número de veces que aparece el término t en el documento d. El cálculo por defecto es $tf_{t,d} = num^{\frac{1}{2}}$.

- idf_t es la frecuencia inversa del documento, calculado tal y como se indica en la ecuación 2.3.
- $coord_{q,d}$, es el factor que indica la presencia de más términos de la consulta en el documento; por lo que se devolverán valores altos cuanto mayor sea el ratio entre q y d, y valores pequeños cuanto menor sea.
- n_q es el factor de normalización usado para poder comparar consultas sobre los mismos documentos. Este factor no afecta al ranking de documentos en una búsqueda, sino que intenta hacer comparables diferentes consultas (incluso de distintos índices). Es un factor calculado en tiempo de búsqueda y se calcula como $n_q = n_\tau = 1/\tau$, donde τ es la suma de los pesos de los cuadrados de los términos de la consulta.
- p_t es el peso asignado en tiempo de búsqueda al término t en la consulta q.
- $n_{t,d}$ encapsula el factor de normalización de longitud y los pesos de los campos del índice asignados a cada uno en tiempo de indexación. Cuando un documento se añade al índice, estos factores se multiplican y almacenan: $n_{t,d} = norm_d \cdot \prod_{\alpha \in d} p_\alpha$.

Cálculo de relevancia

Una vez definido el cálculo de la similitud entre el contexto y los datos primitivos de un nodo, podemos combinar esta medida con la centralidad de grado para obtener la relevancia final de los nodos raíz del grafo, los que representan los conceptos identificados en el contexto. Así, para el cálculo de la relevancia de un nodo n, se multiplica la similitud $sim(\Delta, n_t)$ por el valor de centralidad de grado $C_{DEG}(n)$:

$$\textbf{\textit{relevancia(n)}} = sim(\Delta, n_t) \cdot (1 + \frac{C_{DEG}(n)}{N}) \tag{4.6}$$

Para normalizar el cálculo de la centralidad, dividimos por el número total de nodos, N. En el caso de que un nodo esté aislado dentro del grafo, sumamos 1 a la centralidad para no anular el cálculo de la similitud de dicho nodo con el contexto y mantener así un valor de relevancia. Esto puede suceder, ya que el grafo no tiene por qué ser necesariamente conexo. En el caso de los nodos raíz que no han encontrado camino con ningún nodo hoja, se consideran nodos aislados, pero no eliminables, ya que representan los conceptos obtenidos en el contexto. Si la similitud es cero, su relevancia también lo es, porque independientemente de cómo esté conectado con el resto de nodos, su temática no se relaciona con el contexto de partida.

4.5. Validación de ADEGA

Para hacer una comparativa justa con otros anotadores semánticos del estado del arte, tuvimos que adaptar ADEGA al tipo de resultado que devuelven dichos anotadores. Como se ha explicado, no hay ninguna aproximación que utilice el grafo construido en el proceso de desambiguación o filtrado para generar valor y añadir conocimiento a la solución, ya que sólo se utiliza como un medio para llegar a un fin: una buena anotación. Tampoco los conjuntos de datos que usaremos, y que están disponibles en los principales sistemas de comparación de aproximaciones, contemplan una salida mucho más completa más allá de una entidad por anotación.

Dado que uno de los principales objetivos de ADEGA es realizar una buena anotación de contenido textual, además de enriquecerlo con un grafo de conceptos relacionados, vamos a realizar una comparativa teniendo en cuenta sólo una entidad por anotación. De esta forma, compararemos nuestra aproximación de obtención de contexto, generación del grafo y filtrado de las entidades con otras aproximaciones propuestas.

Debido a que recientemente el desarrollo de sistemas de anotación semántica ha despertado el interés de múltiples grupos de investigación que tratan de extraer datos estructurados de texto plano [15], la comunidad ha invertido cierto esfuerzo en desarrollar sistemas como marco de referencia para realizar comparativas justas entre diferentes aproximaciones, utilizando las mismas medidas y conjuntos de datos [90, 24, 105].

BAT-framework [24] fue el primer sistema ampliamente aceptado por la comunidad. Se basa en la definición de un conjunto de problemas relacionados con la tarea de anotación de entidades, un conjunto de medidas para evaluar el rendimiento de los sistemas y una colección de todos los conjuntos de datos disponibles hasta la fecha. El software, disponible a la comunidad como código abierto, es fácilmente extensible con nuevos anotadores y conjuntos de datos.

Como evolución a este sistema se propuso *GERBIL* [105], que utiliza gran parte de la infraestructura de *BAT-framework*, pero incorpora la noción de comparativa agnóstica en cuanto a base de conocimiento, introduce almacenamiento para guardar los resultados parciales y finales, además de ser un repositorio de resultados de anotación central, de forma que se pueda consultar y realizar pruebas en línea con los sistemas registrados en la plataforma.

Para realizar la comparativa de ADEGA con respecto a otros sistemas desarrollados del estado del arte, hemos escogido *BAT-framework*, debido a que *GERBIL* no acaba de obtener el consenso de la comunidad con respecto a decisiones tomadas al modificar medidas de preci-

sión y cobertura, y a la forma de implementar el mapeo entre los resultados de las propuestas y el sistema agnóstico estándar que utilizan para las comparativas.

4.5.1. Definición de las tareas de anotación de entidades

BAT-framework define una jerarquía de tareas de anotación de entidades en forma de grafo acíclico dirigido, que cubre el amplio espectro de objetivos en las soluciones de anotación que manejan la mayor parte de los sistemas. Esta jerarquía se puede dividir en dos tipos principales, con tres problemas cada uno, que consisten en:

1. La identificación (y posible pesado) de las anotaciones, lo que conlleva a indicar correctamente los pares mención-entidad a los que hacen referencia en el texto.

2. La búsqueda del conjunto de anotaciones correctas de un texto (con posible pesado), sin la necesidad de enlazar la entidad a una mención en el texto.

No todas las propuestas soportan las definiciones más restrictivas para estos tipos de problemas. Lo mismo sucede con los conjuntos de datos, ya que muchos no contienen información tan precisa, como las menciones a las que hace referencia una anotación en el texto de partida. Por este motivo, para realizar una comparativa extensa entre las distintas propuestas y conjuntos de datos del estado del arte, hemos escogido una representación genérica de cada tipo de problema que se ajusta a todas las aproximaciones:

- Para el primer tipo de problema resolveremos **D2W** (del inglés *Disambiguation to Wikipedia*), desambiguación de menciones utilizando la Wikipedia, inicialmente introducido por [25]. El objetivo de este experimento es mapear un conjunto de menciones $m \in M$ del texto de partida a entidades de una base de conocimiento $e \in W$. Como resultado, se devolverá un conjunto de pares mención-entidad $\{(m,e)\}$.

- Para el segundo tipo, resolveremos **C2W** (del inglés *Concepts to Wikipedia*), detección de conceptos utilizando la Wikipedia, donde se identificarán los conceptos con los que se anota el texto de partida. El resultado es un conjunto de entidades $\{e\}$ para el texto de partida. Este problema fue introducido inicialmente por [66].

Hay que puntualizar que C2W es computacionalmente más complejo que D2W, ya que no se parte del conjunto de menciones que hay que anotar, sino que se deja a elección del

Conjunto de datos	Nº doc.	Long. media	Nº anot.	Media anot/doc
AQUAINT	50	1415	727	14.5
MSNBC	20	3316	612	30.6
IITB	103	3879	9043	87.7
ACE2004	57	2300	225	3.9
AIDA/CoNNL Test B	231	1039	4485	19.5
Spotlight	58	173	306	5.3

Tabla 4.4: Características de los conjuntos de datos usados en la validación de ADEGA, donde *Nº doc.* es el número de documentos en el conjunto de datos, *Long. media* es la longitud media de los textos expresada en caracteres, *Nº anot.* es el número total de términos anotados y *Media anot/doc* es la media de anotaciones por documento

algoritmo de anotación seleccionar aquellos términos que son importantes, para después mapearlos con las entidades de la Wikipedia. La base de conocimiento contra la que realizaremos las tareas será la DBpedia, ya que existe una correspondencia uno a uno entre las URL de la Wikipedia y la DBpedia.

4.5.2. Conjuntos de datos utilizados

Para realizar los experimentos, hemos utilizado seis de los siete conjuntos de datos disponibles en *BAT-framework*. La temática del contenido es variada, siendo la mayor parte sobre noticias de política, deportes, actualidad o tecnología. Hemos descartado uno de los conjuntos, concretamente *Meij*, ya que es un conjunto especial compuesto de *tweets*, cuyo contenido es corto (menos de 140 caracteres) y pobre, normalmente utilizando palabras abreviadas o mal escritas. Este tipo de texto no forma parte del objetivo de anotación de ADEGA. La tabla 4.4 contiene en detalle las características de cada uno de los conjuntos de datos. Además:

- *AQUAINT* [71] consiste en un conjunto de textos provenientes de un servicio de noticias. No se anotan todas las ocurrencias de las menciones, sólo la primera mención de cada entidad y las que se consideran más relevantes.

- *MSNBC* [25] es un conjunto de noticias de la red de *MSNBC news*. Anota sólo las entidades más relevantes y sus menciones en el texto.

- *IITB* [56] contiene más de cien textos anotados manualmente, los cuales fueron obtenidos de una web sobre deportes, entretenimiento, ciencia, tecnología y salud. Es el

conjunto de datos más detallado, ya que se anotan la mayor parte de las menciones, incluyendo aquellos conceptos que no son altamente relevantes.

- *ACE2004* [88] es un conjunto de datos sobre noticias, donde se anotan las menciones de los nombres de entidades más relevantes, ignorando los nombres comunes.
- *AIDA/CoNLL Test B* [44] está construido sobre los datos utilizados para la tarea de reconocimiento de entidades de CoNLL 2003. Los documentos son noticias obtenidas de *Reuters Corpus V1*. Se ha anotado un subconjunto de menciones que hacen referencia a nombres de entidades, pero los nombres comunes se han ignorado. Consiste en tres subconjuntos: *Training*, *Test A* y *Test B*. Dado que el sistema AIDA, propuesto en la misma publicación, se ha entrenado utilizando los dos primeros, sólo usaremos el último conjunto para realizar los experimentos.
- *Spotlight* [67] es un conjunto de noticias cortas de actualidad donde se anotan los conceptos más relevantes de los documentos.

A través de estos conjuntos de datos, podemos valorar la eficiencia y rendimiento de ADEGA cuando tiene que anotar diferentes textos con características variadas y donde se deben anotar términos de distinta naturaleza y morfología (nombres, adjetivos, verbos, entidades nombradas, etc). De modo que se puede evaluar ADEGA desde diferentes perspectivas, resaltando en ellas las ventajas y las deficiencias del algoritmo. Además, cabe destacar que estos conjuntos de datos han sido generados en trabajos con los que se comparará ADEGA.

4.5.3. Medidas de precisión y cobertura

Para poder realizar una comparativa justa entre dos sistemas de anotación es necesario definir lo que significa una "coincidencia correcta" entre las anotaciones que se obtienen como solución del algoritmo y las anotaciones objetivo. *BAT-framework* propone generalizar las medidas estándar de verdaderos/falsos positivos, verdaderos/falsos negativos, precisión y cobertura, añadiendo una relación binaria M que especifica la idea de "coincidencia correcta" entre dos anotaciones.

Para los problemas de tipo C2W, M es bastante sencilla, ya que este tipo de problemas considera coincidencias sólo entre dos entidades. Las soluciones de los sistemas de anotación pueden contener URLs de redirección en sus resultados, por lo que hay que considerar el desreferenciarlas para obtener el resultado correcto. Si tenemos en cuenta que las páginas de la

Wikipedia se dividen en dos conjuntos: *R* para las páginas de redirección (consideradas como sinónimos de la página original) y *nR* para las páginas sin redirección, podemos considerar *R* como una relación de muchos-a-uno hacia *nR*. En ese caso, definimos la función $f : R \cup nR \mapsto nR$ y decimos que existe una *coincidencia fuerte entre una entidad* e_1 *y otra entidad* e_2, M_e, si $f(e_1) = f(e_2)$.

En cuanto a los problemas de tipo D2W, la salida consiste en un conjunto de anotaciones que involucran al par mención-entidad, (m,e); por lo tanto, M debe trabajar con este tipo de soluciones. Una mención m es la ocurrencia de un término en el texto y puede codificarse por el par (p,l), siendo p la posición de la mención dentro del texto de partida y l la longitud de la misma. Digamos que tenemos que calcular $M_a(a_1,a_2)$ donde $a_1 = ((p_1,l_1),e_1)$ y $a_2 = ((p_2,l_2),e_2)$; definimos la *coincidencia fuerte entre anotaciones* M_a como una relación binaria que se verifica si $p_1 = p2$, $l_1 = l_2$ y $f(e_1) = f(e_2)$, donde f es la función de desreferencia definida en el párrafo anterior. Hay que tener en cuenta que las menciones vienen establecidas como entrada para realizar anotación en este tipo de problemas.

En el cálculo de la precisión, cobertura y medida-F se incluye M para tener en cuenta las coincidencias entre las anotaciones. Consideramos que s_d y g_d son, respectivamente, la solución encontrada por el anotador y el resultado objetivo para un documento $d \in D$, por lo tanto podemos definir:

$$\begin{aligned} P(s,g,M) &= \frac{|VP(s,g,M)|}{|VP(s,g,M)| + |FP(s,g,M)|} \\ C(s,g,M) &= \frac{|VP(s,g,M)|}{|VP(s,g,M)| + |FN(s,g,M)|} \\ F_1(s,g,M) &= \frac{2 \cdot P(s,g,M) \cdot R(s,g,M)}{P(s,g,M) + R(s,g,M)} \end{aligned} \tag{4.7}$$

En el ámbito de este capítulo, vamos a utilizar las medidas de micro-precisión y micro-cobertura, que se utilizan para calcular precisión y cobertura del sistema sobre los diferentes conjuntos de datos sumando los valores individuales de VP, FP y FN de cada documento. Teniendo en cuenta que S y G son el conjunto de soluciones del algoritmo y los resultados

objetivo respectivamente, la formulación sería la siguiente:

$$\begin{aligned} P_{mic}(S,G,M) &= \frac{\sum_{d\in D}|VP(s_d,g_d,M)|}{\sum_{d\in D}(|VP(s_d,g_d,M)|+|FP(s_d,g_d,M)|)} \\ C_m icro(S,G,M) &= \frac{\sum_{d\in D}|VP(s_d,g_d,M)|}{\sum_{d\in D}(|VP(s_d,g_d,M)|+|FN(s_d,g_d,M)|)} \\ F_1(S,G,M) &= \frac{2\cdot P(S,G,M)\cdot R(S,G,M)}{P(S,G,M)+R(S,G,M)} \end{aligned} \tag{4.8}$$

4.5.4. Configuración de ADEGA

Debido al cambio realizado en la selección de los potenciales términos del contexto con el algoritmo de *n-gramas*, combinado con el índice de etiquetas de la DBpedia, nuestra selección del contexto mejoró notablemente comparado con la aproximación anterior. Lo que hacíamos era delegar una de las tareas más importantes, la selección de los términos candidatos, a un conjunto de reglas programadas manualmente que podían no ser lo suficientemente precisas para seleccionar todos los términos relevantes; incluso podían incurrir en una incorrecta selección de términos producida por una regla mal definida. Con la selección actual de términos, garantizamos que, además de ser un proceso independiente del lenguaje y de cómo se construyen las frases, la identificación de los términos se acota a aquellos que ya existen en el repositorio con el que queremos anotar, evitando la mayor parte de los errores que podían generarse por una mala definición de reglas.

Gran parte del esfuerzo que se realizaba anteriormente se focalizaba en eliminar aquellos términos del contexto que no eran correctos mediante el uso de análisis posteriores de los mismos: concretamente el filtrado del grafo se utilizaba para eliminar el 80 % de los nodos de la exploración, aquellos que no estaban relacionados con el contexto. Con la aproximación que seguimos ahora para construir el grafo partiendo de los nodos raíz (entidades que representan a los términos del contexto) y los nodos hoja (nodos relacionados con el contexto), el grafo que obtenemos es en su mayor parte temáticamente afín a los nodos que teníamos de partida. Por lo tanto, el cometido del filtrado ya no es eliminar la mayor cantidad de nodos que se obtienen en la construcción del grafo, porque inicialmente limitamos esa exploración a aquellos temáticamente similares, sino que ahora pasa a ser una herramienta para asignar una relevancia más significativa a los nodos raíz que representan las anotaciones de los términos del contexto.

Según las pruebas que hemos realizado sobre las soluciones de anotación de ADEGA con

las soluciones objetivo de los conjuntos de datos introducidos anteriormente, el 83 % de las entidades solución ya se obtienen en la detección del contexto y creación de los nodos raíz. Hay que tener en cuenta que a mayores de las entidades solución, existen muchos términos que no son solución y que hay que eliminar para obtener una precisión alta del algoritmo. Es ahí donde el filtrado y cálculo de relevancia de los nodos es un punto clave para establecer un umbral que nos indique qué nodo raíz es una anotación y qué nodo no lo es. En este punto, la configuración que establezcamos para la creación del grafo y la evaluación de los nodos, cobra gran importancia para llevar a cabo una correcta selección de los nodos raíz que generamos de partida. Los resultados que obtendremos sobre los nodos de este filtrado serán las anotaciones finales devueltas como solución.

Para la creación del grafo, es necesario seleccionar las relaciones que se incluirán en la exploración. Más relaciones significan un grafo más rico, pero también más tiempo invertido en la búsqueda de caminos, ya que la explosión de nodos que obtenemos es exponencial a medida que aumentamos el número de relaciones a consierar. En la versión inicial de ADEGA utilizábamos todas las relaciones disponibles y como consecuencia se aumentaba la generación de nodos a medida que profundizábamos en la solución, penalizando el tiempo que invertíamos en la búsqueda. Para realizar las pruebas de selección del conjunto de relaciones buscamos maximizar el tiempo de ejecución del algoritmo y la calidad de la solución, por lo que hay que llegar a un compromiso al escoger la combinación de las relaciones. Además, también hemos considerado el tiempo en la construcción del grafo provocado por el coste de aumentar un nivel más de profundidad en la exploración. Teniendo esto en cuenta, se han realizado experimentos con los siguientes conjuntos de relaciones:

- *C1*, compuesto por las 15.000 relaciones específicas entre entidades y sus inversas.
- *C2*, compuesto por `dcterms:subject`, `rdf:type`, y las 15.000 relaciones específicas entre entidades.
- *C3*, compuesto por `dcterms:subject`, `rdf:type` y las 15.000 relaciones específicas entre entidades, e inversas de éstas.
- *C4*, compuesto por `dcterms:subject`, `skos:broader` e inversas de ambas (sólo relaciones de jerarquía).

En este análisis, no se incluyeron combinaciones como *C1+C4* o *C3+C4*, ya que en la práctica implica considerar el conjunto de relaciones completo, lo que haría aumentar los tiempos de exploración. Según los resultados de las pruebas, que se muestran en la tabla 4.5, la mejor

Relaciones	Nivel exploración	F1	Precisión	Cobertura
C1	4	0,554	0,979	0,387
C2	4	0,519	0,985	0,352
C3	4	0,586	0,977	0,418
C4	4	0,440	0,986	0,283
C1	6	0,587	0,974	0,420
C2	6	0,549	0,979	0,381
C3	6	0,617	0,976	0,451
C4	6	0,461	0,982	0,301

Tabla 4.5: Pruebas de selección para el conjunto de relaciones de la DBpedia

configuración es la obtenida por el conjunto *C3*. La exploración en nivel 6 obtiene una mejora en la medida-F, pero hemos comprobado que empeora significativamente los tiempos en relación a la exploración de nivel 4. Por lo tanto, hemos descartado el nivel 6, ya que el coste en tiempo no justifica el $0,031\,\%$ que obtenemos de mejora en la medida-F.

En relación al pesado del grafo, no hemos limitado la experimentación al cálculo de la relevancia que definimos en la ecuación 4.6, sino que hemos implementado los algoritmos del estado del arte *PageRank* y *HITS* como método sustitutivo del cálculo de la centralidad de grado.

4.5.5. Resultados

Para realizar la comparativa, hemos seleccionado aquellas propuestas del estado del arte cuyo código fuente o servicio web estuvieran disponibles para ejecutar las pruebas desde nuestra máquina. Nos hemos comparado con *AIDA* [44], cuyo código fuente está disponible para su ejecución en local; y *TagMe2* [32], *DBpediaSpotlight* [67], *AGDISTIS* [104] y *Babelfy* [73], que disponen de servicios web que se pueden consultar para realizar la anotación.

Hemos ejecutado las tareas C2W y D2W para los seis conjuntos de datos. Hay que tener en cuenta que para la tarea C2W, BAT-framework trata de obtener el mejor resultado para cada propuesta en cada conjunto de datos. Para conseguirlo, una vez realizada la prueba, itera sobre los resultados estableciendo un umbral máximo sobre el valor de relevancia de las anotaciones. De esta forma, devuelve los resultados con un umbral máximo de relevancia que maximice el valor de la medida-F en cada conjunto. Este valor del umbral es el indicado en la columna *Mejor t* en la tabla comparativa. En D2W no se establece un umbral de relevancia

máxima, ya que los términos que hay que anotar vienen dados como parámetros de entrada al inicio de la ejecución, no dejando al algoritmo escoger qué términos anotar. También hay que destacar que la propuesta de *AGDISTIS* sólo se ha ejecutado para la tarea D2W, ya que su software no incluye detección de términos a anotar, primera parte de la tarea C2W.

En la tabla 4.6 se pueden ver los resultados de la ejecución de la tarea C2W. ADEGA consigue los mejores resultados en cuatro de seis conjuntos de datos, concretamente en *IITB*, *MSNBC*, *ACE2004* y *Spotlight*; obtiene el segundo mejor resultado en *AQUAINT*, con una diferencia de 12,5 puntos en la medida-F con respecto a la mejor propuesta, *TagMe2*; y el tercer mejor resultado en *AIDA/CoNLL Test B*, con una diferencia de 6,8 puntos en la medida-F respecto al mejor, *AIDA*. Analizando las consecuencias que provocan el bajo valor de F1 en estos conjuntos, comprobamos que en el caso de *AQUAINT*, la mitad de las anotaciones corresponden a nombres de entidades y la otra mitad a nombres comunes, donde radica el problema. Son términos poco representativos, cuya relevancia es mínima porque no son términos importantes en el texto, aunque el conjunto de datos los considere términos de anotación. ADEGA les asigna un peso bajo con respecto al resto del contexto y a mayores no se conectan con los términos que conforman el grafo. Debido al poco peso que consiguen en la fase de pesado y la falta de conexiones en el grafo, son términos que aunque formen parte de la solución, no consiguen llegar al umbral requerido que maximiza la F1 y se descartan. En el caso de *AIDA/CoNLL Test B*, un conjunto de datos que anota nombres de entidades, se pone de manifiesto la necesidad de disponer de un contexto rico que nos permita eliminar aquellas entidades que no son las correctas. En la tabla 4.7 analizamos los resultados obtenidos por ADEGA para el conjunto calculando, para un rango determinado de F1, el número de documentos que se encuentran en cada rango, la longitud media del texto, el número de anotaciones medias y la relación que hay entre la longitud del texto y las anotaciones. Esta relación indica la densidad de términos que hay que anotar. Cuanto menor sea la relación, mayor densidad de anotaciones; que pueden ser debidas a textos muy cortos o textos de longitud media, pero con una gran densidad de términos. En el caso de tener una relación mayor, la densidad disminuye, correspondiéndose a textos más largos con menos anotaciones. Un 26 % de los documentos que tienen menos de 0,5 de F1, son documentos un 24 % más cortos, con un 35 % más de anotaciones medias que el 74 % de documentos restante. Se corresponden con textos cortos con muchas anotaciones o documentos de tipo listas de apellidos (sin nombres), donde es complicado generar un contexto que nos permita realizar una correcta desambiguación. En este caso, son documentos que tienen una relación entre longitud media y anotaciones de 37,6, en relación al 58,3 restante.

Datos	Anotador	Mejor t	$F1_{micro}$	P_{micro}	R_{micro}
IITB	ADEGA	0,000	52,0	43,2	65,4
	ADEGA PageRank	0,000	52,0	43,2	65,4
	ADEGA HIT	0,000	46,7	52,6	42,0
	TagMe 2	0,117	45,9	45,8	46,1
	DBpedia Spotlight	1,000	37,1	33,9	40,9
	Babelfy	0,000	39,1	29,1	59,8
	AIDA	0,141	19,0	63,4	11,1
AQUAINT	TagMe 2	0,188	64,5	63,2	65,9
	ADEGA	0,117	52,0	52,3	51,8
	ADEGA HIT	0,000	47,6	39,7	59,4
	ADEGA PageRank	0,031	40,6	34,2	49,7
	DBpedia Spotlight	1,000	44,9	43,8	46,0
	AIDA	0,250	46,4	66,4	35,6
	Babelfy	0,000	27,7	17,3	69,2
MSNBC	ADEGA	0,102	72,2	84,3	63,1
	ADEGA PageRank	0,000	70,7	80,3	63,1
	AIDA	0,172	66,2	75,6	58,9
	TagMe 2	0,328	62,7	65,5	60,1
	ADEGA HIT	0,008	47,0	35,7	68,8
	DBpedia Spotlight	0,055	30,4	21,3	53,4
	Babelfy	0,000	19,4	11,0	79,2
ACE2004	ADEGA	0,148	44,0	31,7	71,7
	ADEGA PageRank	0,016	43,8	31,4	72,2
	AIDA	0,133	38,3	25,6	76,2
	Tagme 2	0,312	31,6	22,0	56,1
	DBpedia Spotlight	1,000	12,7	7,2	53,8
	ADEGA HIT	0,961	4,6	15,4	2,7
	Babelfy	0,156	2,6	60,0	1,3
AIDA/CoNLL Test B	AIDA	0,141	66,6	73,3	61,1
	Tagme 2	0,227	64,3	58,2	71,7
	ADEGA	0,070	59,8	68,0	53,3
	ADEGA PageRank	0,000	59,5	67,0	53,5
	ADEGA HIT	0,000	44,3	37,2	54,7
	DBpedia Spotlight	0,992	41,5	29,2	71,9
	Babelfy	0,008	26,3	31,9	22,3
Spotlight	ADEGA	0,016	60,4	55,2	66,8
	ADEGA PageRank	0,000	60,3	55,0	66,8
	ADEGA HIT	0,000	53,3	60,3	47,8
	DBpedia Spotlight	0,914	49,8	47,1	52,9
	Tagme 2	0,047	47,7	39,3	60,7
	Babelfy	0,000	43,3	34,6	58,0
	AIDA	0,000	20,2	59,0	12,2

Tabla 4.6: Comparativa C2W

Rango F1	Nº doc.	long media	anot. medias	Relación long/anot. media
0 - 0,49	60	856,9	22,8	37,6
0,5 -1	171	1126,7	19,3	58,3

Tabla 4.7: Análisis de los resultados obtenidos por ADEGA en el conjunto de datos *AIDA-CoNLL Test B*

Cuanto menor es la relación, más difícil es para ADEGA generar una buena anotación.

En la tabla 4.8, podemos ver los resultados de la ejecución de la tarea D2W. ADEGA consigue los mejores resultados en cuatro de los seis conjuntos: *IITB*, *AQUAINT*, *MSNBC* y *Spotlight*; en el conjunto *ACE2004* obtiene una diferencia en la medida-F de 0,4, poco significativa con respecto a *TagMe2*, quedándose en tercera posición; mientras que en *AIDA/CoNLL Test B* obtiene el cuarto mejor resultado, con una diferencia de 11 puntos en la medida-F con respecto al primero, *DBpediaSpotlight*. El análisis llevado a cabo sobre estos conjuntos de datos, evidencia el mismo problema en el conjunto *AIDA Test B* que se describe en el párrafo anterior para la ejecución de la tarea C2W. Por las características de este conjunto de datos, mucha densidad de anotaciones en textos cortos o tipo lista con bajas probabilidades de generar un contexto que permita una buena desambiguación, repercute negativamente en el resultado de F1 conseguido, reduciendo el 75 % de F1 conseguida en el 74 % de los documentos al 63 % global que se consigue.

En la tabla resumen 4.9, que contiene el cálculo del valor promedio de la medida-F para todos los conjuntos de datos, ADEGA obtiene el mejor F1 promedio en las tareas C2W y D2W, con un valor de 56,73 % y 80,73 %, respectivamente. El segundo mejor resultado lo obtiene *TagMe2*, con 52,78 % en C2W y 72,72 % en D2W; seguido de *DBpediaSpotlight*, con 36,07 % y 70,85 %, respectivamente; *AGDISTIS*, con 58,88 % en D2W; *Babelfy*, con 26,4 % y 54,3 %; y finalmente *AIDA*, con 42,78 % y 49,3 %. También hemos comprobado que la medida de evaluación de nodos presentada en la sección 4.4.2 ofrece unos mejores resultados que los obtenidos por los algoritmos *PageRank* y *HITS*, con una media de F1 de 54,4 % y 40,6 % en C2W, respectivamente; y 78,05 % en D2W por ambos. Aun así, ambas aproximaciones consiguen mejorar las propuestas del estado del arte.

Datos	**Anotador**	$F1_{micro}$	P_{micro}	R_{micro}
IITB	ADEGA	87,8	92,0	84,1
	ADEGA PageRank	87,8	92,0	84,1
	ADEGA HIT	87,8	92,0	84,1
	TagMe 2	66,6	73,4	60,9
	DBpedia Spotlight	60,3	66,7	55,0
	AGDISTIS	47,2	69,8	35,7
	Babelfy	56,3	66,8	48,6
	AIDA	19,6	68,8	11,4
AQUAINT	ADEGA PageRank	86,3	93,5	80,1
	ADEGA HIT	86,3	93,5	80,1
	ADEGA	85,7	93,4	79,2
	TagMe 2	77,6	79,9	75,4
	DBpedia Spotlight	76,6	81,8	72,0
	AGDISTIS	61,3	75,5	51,6
	Babelfy	55,0	59,7	51,0
	AIDA	47,0	75,6	34,1
MSNBC	ADEGA	83,0	84,9	81,2
	TagMe 2	76,7	82,9	71,4
	AGDISTIS	76,0	78,3	73,9
	DBpedia Spotlight	68,7	77,3	61,8
	ADEGA HIT	66,1	75,1	59,0
	ADEGA PageRank	66,1	75,1	59,0
	AIDA	62,9	70,1	57,1
	Babelfy	49,5	53,4	46,1
ACE2004	Tagme 2	81,6	84,7	78,7
	ADEGA PageRank	81,5	91,7	73,3
	ADEGA HIT	81,5	91,7	73,3
	AIDA	81,7	86,6	77,3
	ADEGA	81,2	91,6	72,9
	DBpedia Spotlight	78,7	80,2	77,3
	AGDISTIS	72,3	76,6	68,4
	Babelfy	59,0	62,4	56,0
AIDA/CoNLL Test B	DBpedia Spotlight	74,0	75,5	72,6
	Tagme 2	70,5	75,0	66,6
	AIDA	64,9	68,1	61,9
	ADEGA	63,0	70,2	57,2
	ADEGA PageRank	63,0	70,2	57,2
	ADEGA HIT	63,0	70,2	57,2
	AGDISTIS	58,2	62,9	54,2
	Babelfy	50,8	55,8	46,6
Spotlight	ADEGA	83,7	89,0	79,1
	ADEGA HIT	83,6	88,6	79,1
	ADEGA PageRank	83,6	88,6	79,1
	DBpedia Spotlight	66,8	75,6	59,8
	TagMe 2	63,3	66,0	60,8
	Babelfy	55,2	60,2	51,0
	AGDISTIS	38,3	71,4	26,1
	AIDA	19,7	61,0	11,8

Tabla 4.8: Comparativa D2W

	Media $F1_{micro}$	
	C2W	D2W
ADEGA	56,73	80,73
ADEGA PageRank	54,48	78,05
ADEGA HIT	40,58	78,05
TagMe 2	52,78	72,72
DBpedia Spotlight	36,07	70,85
AGDISTIS	-	58,88
Babelfy	26,4	54,3
AIDA	42,78	49,3

Tabla 4.9: Promedio de la medida-F de las ocho propuestas de anotación para todos los conjuntos de datos.

4.6. Conclusiones

Aunque la primera implementación de ADEGA inicialmente fuese muy prometedora en términos de precisión y cobertura, el rendimiento que alcanzábamos con la solución era pobre y difícilmente aplicable a un problema real, donde las respuestas a las necesidades de información se exigen prácticamente en tiempo real. La posterior paralelización del algoritmo nos ha servido para afrontar un problema de anotación sobre grandes volúmenes de datos, pero no habíamos atacado el problema de raíz, ya que seguíamos obteniendo tiempos prohibitivos a la hora de anotar un único término. La versión de ADEGA que se presenta en este capítulo, soluciona en gran medida estos problemas, manteniendo la calidad de las soluciones.

Hemos detectado la necesidad de procesar parte de la información de partida para agilizar los tiempos de respuesta y de cambiar las tecnologías que utilizábamos para mejorar el almacenamiento hacia otro más eficiente. Debido al cambio de perspectiva de una exploración exhaustiva del grafo hacia un problema de búsqueda de caminos, hemos acotado el espacio de búsqueda y reducido la complejidad para crear un algoritmo mucho más eficiente. Aplicando técnicas de teoría de grafos, pudimos incorporar en una sola fórmula la estructura de la red de nodos y la similitud textual para obtener un valor de relevancia de las entidades con las que anotamos los términos relevantes de un texto.

Adicionalmente, hemos comparado nuestra propuesta de anotación de entidades con soluciones del estado del arte. Debido a que hasta la fecha no se ha presentado ninguna aproximación donde el grafo forme parte de la solución, hemos validado el algoritmo hasta la

selección correcta de la entidad raíz, ya que no hemos conseguido ni conjuntos de datos independientes, ni propuestas tan completas con las que comparar nuestra creación de grafos para el enriquecimiento de la solución.

Como resultado de la comparativa, se concluye que ADEGA mejora el estado del arte en anotación semántica usando la DBpedia. La segunda mejor aproximación, *TagMe2*, obtiene peores resultados en 4 de los seis conjuntos analizados para la tarea C2W, al igual que en D2W.

Capítulo 5

Anotación semántica del repositorio de Universia

5.1. Introducción

ADEGA se ha aplicado a diferentes problemas de etiquetado, enriquecimiento y clasificación de contenidos. Tal y como se ha comentado en el capítulo 2, se ha utilizado para enriquecer los contenidos de aprendizaje de cursos de Geografía e Historia en educación primaria, facilitando el acceso de los estudiantes a información adicional sobre tópicos que no habían sido cubiertos convenientemente en dichos contenidos. La validación descrita en el capítulo 2 avala la calidad de los resultados obtenidos con la primera versión de ADEGA para el enriquecimiento de documentos.

La nueva versión de ADEGA (capítulo 4) también ha sido aplicada en dos proyectos de clasificación y etiquetado de textos en el ámbito educativo, en los que se exige un mayor rendimiento del anotador y donde la primera versión de ADEGA no ofrecía un rendimiento suficiente para dar respuesta a las exigencias de anotación de textos. Por una parte, ADEGA se usó para etiquetar el contenido de las entradas de blogs generadas por los estudiantes en el contexto de una red social de aprendizaje en la materia de *Tecnología Educativa* del Grado de Pedagogía de la Universidad de Santiago de Compostela [35]. El objetivo de este etiquetado fue analizar en qué medida las entradas de blog generadas durante el curso (cerca de 500) tienen un contenido que está relacionado con tópicos de la materia y, por tanto, son contribuciones valiosas de los estudiantes. Como resultado de esta anotación se concluyó que cerca de

un 90 % de las entradas de blog estaban directamente relacionadas con tópicos y contenidos de la materia. En este experimento, la anotación de ADEGA a través de grafos semánticos permitió etiquetar las entradas de blog con conceptos que no estaban directamente vinculados a términos del texto, sino que se relacionaban con los términos que sí aparecían a través de las entidades contenidas en los grafos.

Por otra parte, la nueva versión de ADEGA también se ha aplicado a la anotación de recursos educativos que introducen los usuarios en *Edu-AREA* [13], una plataforma para la gestión de unidades de aprendizaje que se comparten como recursos educativos para que puedan ser reutilizados por profesores en diferentes contextos educativos. Así, a partir del título del recurso y de su descripción, ADEGA sugiere en tiempo real un conjunto de palabras clave con las que se etiqueta cada recurso [14] y que pueden ser modificadas y/o eliminadas por los usuarios si consideran que con ellas no se representan los tópicos de los recursos. Teniendo esto en cuenta, se realizó un experimento en el que participaron 25 usuarios que introdujeron 80 recursos educativos, cada uno de los cuales se etiquetó con 5 palabras clave. Los resultados del experimento demostraron que, por lo general, los usuarios aceptan entre 3 y 4 palabras sugeridas por ADEGA, confirmándose como un servicio valioso a la hora de soportar la edición de recursos educativos abiertos.

En estos proyectos, ADEGA se ofreció como servicio de anotación a una serie de aplicaciones con necesidades de etiquetado de textos generados por usuarios en el marco de redes sociales y/o plataformas educativas. Sin embargo, con el objetivo de usar ADEGA para crear repositorios enlazados con la DBpedia, se creó un repositorio semántico con una porción de más de 60.000 objetos de aprendizaje de la biblioteca de Universia, los cuales se anotaron y enlazaron con las clasificaciones de la DBpedia. En este capítulo se describe en detalle la infraestructura creada para desarrollar este repositorio semántico.

5.2. Representación de objetos de aprendizaje

En la última década se ha realizado un gran esfuerzo para desarrollar estándares que representen las características (o metadatos) de los objetos de aprendizaje (LOs) con el objetivo de facilitar su uso, intercambio y reutilización entre las diferentes herramientas educativas [108]. Hay dos estándares que sobresalen sobre otros, que son el estándar Dublin Core [43], orientado a modelar cualquier recurso digital; y el IEEE LOM [23] desarrollado específicamente para representar las características pedagógicas de los recursos utilizados en actividades de

aprendizaje.

En paralelo a este esfuerzo, el desarrollo de repositorios de LOs propició la aparición de nuevas infraestructuras y arquitecturas mejor adaptadas a la necesidades de esta tecnología, así como nuevos protocolos para la publicación, consulta y reutilización de los LO a través de sus metadatos [64, 100]. Estos repositorios se pueden clasificar desde varias perspectivas [65], entre las que destacamos: (*i*) los que contienen LOs genéricos que cubren una serie de temas o ámbitos pedagógicos diferentes; y (*ii*) los que modelan la semántica de los metadatos de los LOs a través de ontologías. Este último tipo de repositorios, llamados repositorios semánticos, son particularmente interesantes porque mejoran la interoperabilidad entre las herramientas educativas, soportan consultas semánticas sobre el contenido del repositorio; y facilitan la generación automática o el enriquecimiento de algunos metadatos [17]. Este capítulo se centra precisamente en este último aspecto y específicamente en cómo las categorías de los LOs pueden ser automáticamente generadas o anotadas a través de ontologías de forma automática.

La descripción de los metadatos a través de ontologías ha atraído una gran atención gracias al desarrollo de técnicas para la generación automática de esta información, por lo que han aparecido muchas soluciones para tratar de anotar los metadatos de los LOs [85, 50, 27, 4, 78]. Aun así, estas propuestas del estado del arte comparten una serie de problemas comunes: (*i*) usan ontologías que describen la semántica de un dominio particular, haciendo imposible aplicar estas propuestas a la creación o mantenimiento de un repositorio multi-temático de objetos; (*ii*) y llevan a cabo un proceso de anotación semi-automático, donde se requiere de la intervención de los usuarios para anotar y validar la exactitud de los metadatos. Otras propuestas trataron este último problema automatizando el proceso de anotación de LOs [11, 81, 98]. Sin embargo, desde la perspectiva de la anotación de las categorías de los objetos, estas soluciones tienen un problema importante: buscan una única entidad en la ontología para categorizar el objeto. Encontrar exactamente esa entidad y que sea única es demasiado optimista. Dado que esto es poco probable, nuestro enfoque se centra en la extracción de los términos relevantes del LO, en la búsqueda de las mejores entidades que caracterizan a estos términos en la DBpedia; y en la extracción de las categorías más adecuadas que clasifican a estas entidades, permitiéndonos clasificar también al LO.

Por lo tanto, en este capítulo describimos una propuesta para clasificar los objetos de aprendizaje con un conjunto de categorías extraídas automáticamente de la DBpedia. El objetivo es relacionar el contenido textual de los campos del IEEE LOM que describen la temática del objeto, almacenados en el repositorio digital de Universia [3], con un conjunto de cate-

gorías descritas semánticamente en la DBpedia.

5.3. Repositorio de objetos de aprendizaje de Universia

El repositorio de Universia está compuesto por 213 colecciones de objetos de aprendizaje en diferentes idiomas y pertenecientes a diferentes campos científicos. En total, estas colecciones consisten en 15.750.979 LO que sirven como recursos académicos, entre los que se encuentran tesis doctorales, publicaciones científicas y revistas. Los metadatos de estos recursos están representados siguiendo el estándar IEEE LOM y clasificados a partir de la clasificación UNESCO. Sin embargo, la mayor parte de los LOs no se encuadran dentro de ninguna categoría UNESCO, por lo que son recursos que no están clasificados, o en el mejor de los casos, clasificados en temáticas no estándar introducidas manualmente por las personas a cargo de la creación de las instancias LOM. Sólo una pequeña parte de los objetos están correctamente clasificados, en concreto, un 1 % del contenido de Universia; lo que dificulta enormemente la búsqueda y recuperación de los objetos, incrementando el tiempo necesario para encontrar un recurso concreto.

Nuestra propuesta trata de resolver este problema, proporcionando automáticamente una clasificación apropiada para los objetos. El mecanismo de clasificación propuesto usa las categorías que proporciona la Wikipedia a través de la ontología de la DBpedia. Por lo tanto, todos los objetos del repositorio estarán clasificados de forma automática y uniforme, complementando la clasificación existente en los objetos que ya tengan una clasificación UNESCO y proporcionando un mecanismo de clasificación fiable a aquellos LO no científicos ni tecnológicos.

5.4. Enlazando objetos de aprendizaje con DBpedia

El estándar IEEE LOM define un esquema compuesto por siete elementos que permiten representar metadatos sobre un LO: información genérica sobre el recurso (`General`), información técnica (`Technical`) y pedagógica (`Educational`); sus autores y diferentes versiones; su ciclo de vida (`Lifecycle`); sus condiciones de uso y los derechos de propiedad intelectual (`Rights`); sus relaciones con otros objetos (`Relation`); su temática en relación a un sistema de clasificación particular (`Classification`); los comentarios relacionados con su uso educacional (`Annotation`); e información sobre los propios metadatos (`Meta-Metadata`). La figura 5.1 muestra parte de los metadatos de un LO almacenado en

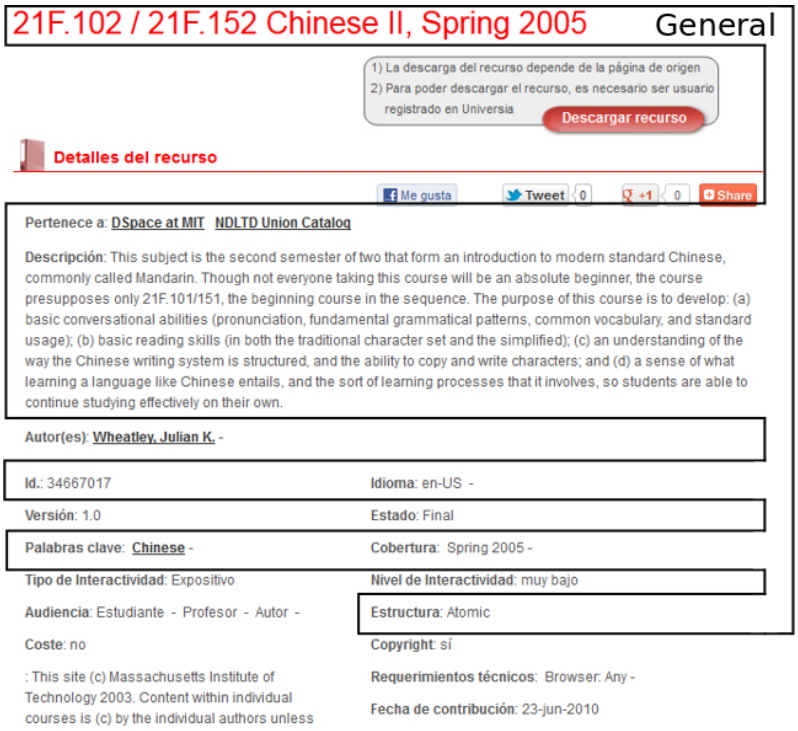

Figura 5.1: Ejemplo de los metadatos de un objeto de aprendizaje recuperado de la biblioteca de Universia

Universia sobre el idioma chino, en la imagen se resalta la información que corresponde al elemento `General` del objeto.

Con el fin de integrar los objetos de aprendizaje de Universia con la DBpedia, es necesario traducir la especificación LOM a una ontología. Para este propósito, usamos la sección LOM de la ontología ISM LD descrita en [5]. La figura 5.2 muestra parte de la ontología, donde los conceptos `General` y `Classification` (coloreados en gris) son los elementos del estándar LOM que pueden ser semánticamente enlazados con relaciones hacia repositorios externos. El objetivo es relacionar dos atributos del concepto `Taxonpath` con la DBpedia: `taxon`, que especifica la categoría; y `source`, que establece el sistema de clasificación usado. Una vez que se selecciona una categoría de la DBpedia para clasificar el objeto, se siguen los siguientes pasos para añadirlo a `Taxonpath`:

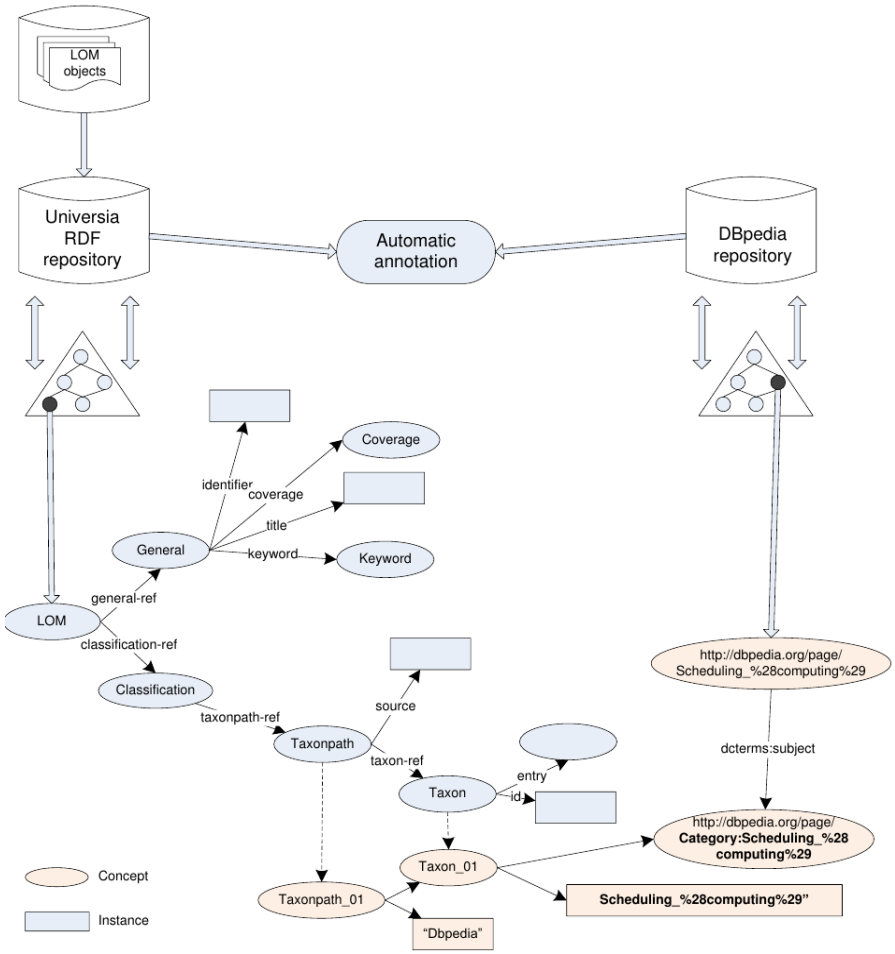

Figura 5.2: Enlazado entre un objeto de aprendizaje del repositorio y la DBpedia a través de la relación *dcterms:subject*

Elemento LOM	Descripción	Frecuencia en GLOBE	Frecuencia en Univesia
`Title`	Nombre de un LO dado	1.00	0.99
`Description`	Descripción textual del contenido del LO	0.87	0.73
`Keywords`	Lista de términos que describen la temática del LO	0.57	0.68
`Coverage`	Características temporales, culturales y geográficas	0.10	0.13

Tabla 5.1: Frecuencia de uso de los elementos IEEE LOM en GLOBE y Universia.

- El valor del atributo `source` es *DBpedia*, porque es el sistema de clasificación usado para asignar las categorías a los objetos de aprendizaje.
- El atributo `taxon` tiene dos relaciones: la relación `id`, cuyo valor es la URI de la categoría de la DBpedia; y la relación `entry`, cuyo valor es la etiqueta (o nombre) de la categoría de la DBpedia, proporcionada por `skos:prefLabel`.

5.4.1. Descripción de la propuesta

Para poder obtener las categorías de la DBpedia para un objeto de aprendizaje, es necesario procesar el contenido de los metadatos para extraer los términos relevantes. Sin embargo, no todos los elementos de LOM contienen información valiosa para poder obtener los términos. Por ejemplo, no todos los campos del elemento `Thecnical` y alguno de los campos de `Lifecycle` proporcionan información útil para determinar la temática de un LO. Por lo tanto, nos hemos centrado en alguno de los campos del elemento `General`: el `title`, que contiene el nombre del LO; `description`, donde se describe brevemente el contenido; las `keywords`, donde se listan las palabras que resumen la temática del LO; y el campo `coverage` que proporciona información sobre características temporales, geográficas o culturales.

Teniendo esto en cuenta, la arquitectura conceptual de nuestra aproximación se representa en la figura 5.3. La solución recibe como entrada los metadatos del LOM, en concreto los atributos del elemento `General`, con la información descriptiva sobre el LO; y procesa esta entrada en dos pasos:

- *Obtención de los términos relevantes del LO*. En este paso se combinan técnicas de procesamiento de lenguaje natural con técnicas de similitud para determinar los términos relevantes que caracterizan el LO, obteniendo como salida el *contexto* del objeto (ver sección 2.2 para más detalles), como se resume en la tabla 5.2. La relevancia de cada término dependerá de donde esté localizado en la estructura LOM. Por ejemplo, si un

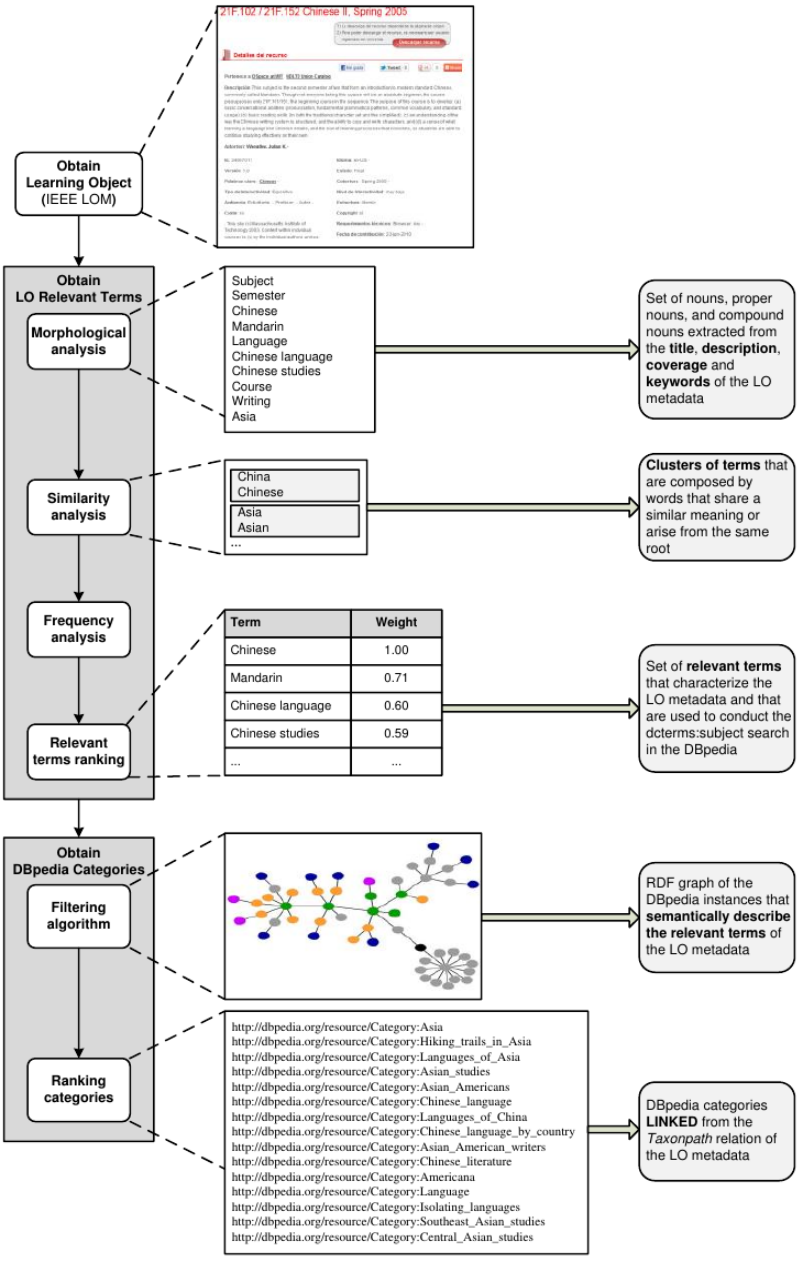

Figura 5.3: Proceso de la tarea de anotación para obtener las categorías de la DBpedia que anotan el elemento *Taxonpath* del IEEE LOM

Término	Entidad de la DBpedia	Relevancia
Chinese	http://dbpedia.org/resource/Chinese	1.00
Mandarin	http://dbpedia.org/resource/Mandarin_Chinese	0.71
Chinese language	http://dbpedia.org/resource/Chinese_language	0.60
Chinese studies	http://dbpedia.org/resource/Sinology	0.59
Asian American studies	http://dbpedia.org/resource/Asian_American_studies	0.59
Course	http://dbpedia.org/resource/Course	0.59
Writing	http://dbpedia.org/resource/Writing	0.50
Asia	http://dbpedia.org/resource/Asia	0.42
Literature	http://dbpedia.org/resource/Literature	0.40

Tabla 5.2: Contexto en inglés obtenido para el LO *Idioma chino*.

término es parte del nombre del LO, tiene sentido que sea más relevante que uno que aparece en la descripción. El peso de cada elemento LOM se ha establecido de acuerdo a lo propuesto por [82], que llevó a cabo un análisis estadístico de la frecuencia de uso de cada elemento en un conjunto de repositorios que pertenecen a la alianza GLOBE, que contienen 630.317 LOs. Como se puede ver en la tabla 5.1, la frecuencia de los campos del elemento `General` para el repositorio GLOBE y para el de Universia es muy parecida; por lo tanto, asumimos que el peso obtenido para GLOBE se puede generalizar para nuestro repositorio.

- *Obtención de las categorías de la DBpedia*. En este segundo paso, usamos los términos relevantes de los metadatos de los LOs para obtener el conjunto de nodos del grafo de la DBpedia que describe semánticamente cada uno de los términos a través de un grafo (ver sección 2.3). Varios de estos nodos son categorías de la DBpedia, que se identifican y ordenan para ser enlazadas al elemento `Taxonpath` del LO, como se puede comprobar en la tabla 5.3.

5.5. Comparativa

En términos de precisión y cobertura, conseguimos los mismos resultados a los obtenidos en la sección de validación 2.4, considerándose buenos resultados para el dominio de aplicación para el que lo estamos aplicando. En esta propuesta de anotación de Universia, el objetivo es recuperar las categorías más adecuadas para un LO, siendo la primera categoría de mayor relevancia la que tiene más probabilidad de ser la que mejor describe al LO. Por ejemplo, las

Categoría de la DBpedia	Entidad	Relevancia
Asia	http://dbpedia.org/resource/Category:Asia	0.112
Hiking trails in Asia	http://dbpedia.org/resource/Category:Hiking_trails_in_Asia	0.112
Languages of Asia	http://dbpedia.org/resource/Category:Languages_of_Asia	0.067
Asian studies	http://dbpedia.org/resource/Category:Asian_studies	0.067
Asian Americans	http://dbpedia.org/resource/Category:Asian_Americans	0.044
Chinese Language	http://dbpedia.org/resource/Category:Chinese_language	0.043
Languages of China	http://dbpedia.org/resource/Category:Languages_of_China	0.043
Chinese language country	http://dbpedia.org/resource/Category:Chinese_language_country	0.039
Asian American writers	http://dbpedia.org/resource/Category:Asian_American_writers	0.038
Chinese literature	http://dbpedia.org/resource/Category:Chinese_literature	0.038

Tabla 5.3: Categorías de la DBpedia asocidadas al LO *Idioma chino*.

	Conjunto 1	**Conjunto 2**	**Conjunto 3**	**Conjunto 4**
RelFinder				
P	0,3626	0,3692	0,1897	0,2464
R	0,2654	0,3708	0,2762	0,5142
F_1	0,2534	0,3700	0,2249	0,3331
$ADEGA(\delta_d{=}2)$				
P	0,3900	0,3779	0,2754	0,2800
R	0,7444	0,7269	0,6695	0,6808
F_1	0,5119	0,4996	0,3903	0,3968

Tabla 5.4: Comparación de RelFinder y ADEGA (dos niveles de profundidad) usando 40 LO de Universia.

categorías de la tabla 5.3 están muy relacionadas con el LO *Idioma chino*, excepto la segunda recuperada.

Adicionalmente, hemos comparado nuevamente nuestra aproximación, obteniendo buenos resultados para valores altos de precisión. La comparativa la hemos llevado a acabo con *Relfinder*, ya que otros algoritmos de anotación del estado del arte disponibles en su momento, como *DBpediaSpotlight* [67], no realizaban categorización sobre el texto. En la tabla 5.4 se pueden ver los resultados obtenidos de la comparativa, que confirman que mejoramos la anotación realizada por esta aproximación, igual que en la sección 2.4.3. La comparativa se llevó a cabo con cuatro conjuntos de test, cada uno compuesto por 10 objetos de aprendizaje seleccionados aleatoriamente de Unviersia; además, se comparó con el mismo número de entidades, limitando el número de soluciones a las devueltas por *Relfinder* para no sesgar los resultados. En la tabla 5.5 se pueden ver los resultados de precisión y cobertura para 40

objetos de aprendizaje del repositorio seleccionados aleatoriamente; los mejores resultados de la medida-F están resaltados en gris, en este caso ADEGA mejora a *Relfinder* en todos los ejemplos menos en siete.

5.6. Implementación de la solución

Siguiendo los principios de los datos enlazados, hemos desarrollado un repositorio semántico que contiene los objetos de aprendizaje de Universia. En concreto, esta implementación está compuesta de:

- Un conjunto de datos que contiene más de 60.663 objetos de aprendizaje descritos en LOM. La versión actual está principalmente compuesta por objetos de temática económica, física o matemática entre otras. Cada objeto se identifica por una URI pública y se enlaza automáticamente a un conjunto de categorías basadas en SKOS de la DBpedia. La tabla 5.6 resume el número de nodos y categorías recuperados para alguno de los objetos analizados, siendo 28,70 el número medio de categorías por objeto.

- Un servicio de búsqueda que permite a los usuarios buscar los objetos de aprendizaje relacionados con una palabra clave. En este caso, relacionado significa que la palabra clave de la consulta coincide o con el título (`title`), la descripción (`description`) o las palabras clave (`keywords`) del elemento `General`; o una categoría de `Taxon`. Se puede ver un ejemplo de la aplicación llevada a cabo para realizar las búsquedas en la figura 5.4.

- Un punto de acceso SPARQL a través del cual los usuarios pueden hacer consultas y recuperar los resultados en varios formatos. Por ejemplo, la figura 5.5 muestra una captura de una consulta donde se solicitan las categorías del LO *Idioma chino*, mientras que la figura 5.6 se presenta el resultado obtenido como solución de la consulta SPARQL. Aunque estos resultados están listados en formato HTML, las soluciones pueden exportarse en otros formatos como RDF/XML o JSON.

LO ID	ADEGA			Relfinder		
	P	*R*	*F*	*P*	*R*	*F*
lo_21855	0.65	0.86	0.74	1.00	0.20	0.33
lo_34666061	0.65	0.81	0.72	0.83	0.31	0.45
lo_34664545	0.30	0.40	0.34	0.40	0.13	0.20
lo_34668373	0.20	0.25	0.22	0.50	0.18	0.27
lo_21580	0.55	0.73	0.73	0.00	0.00	NaN
lo_21594	0.40	0.66	0.50	0.54	0.50	0.52
lo_34664407	0.25	1.00	0.40	0.30	0.60	0.40
lo_34665275	0.35	0.77	0.48	0.50	0.22	0.30
lo_34665427	0.25	1.00	0.40	0.16	0.20	0.18
lo_34665746	0.10	0.66	0.17	0.37	1.00	0.54
lo_34666388	0.45	1.00	0.62	0.04	0.11	0.06
lo_34666494	0.30	0.50	0.37	0.33	0.41	0.37
lo_34666496	0.45	0.90	0.60	0.60	0.60	0.60
lo_34667209	0.20	0.44	0.27	0.16	0.33	0.22
lo_34667236	0.45	0.60	0.51	0.21	0.33	0.26
lo_34667992	0.57	1.00	0.73	0.66	0.36	0.47
lo_34667936	0.50	0.75	0.60	0.16	0.50	0.25
lo_34665989	0.30	0.54	0.38	0.33	0.18	0.23
lo_34667342	0.30	1.00	0.46	0.16	0.16	0.16
lo_21557	0.45	0.90	0.60	0.00	0.00	NaN
lo_34666949	0.25	0.83	0.38	0.25	0.16	0.20
lo_34667225	0.30	0.60	0.40	0.22	0.20	0.21
lo_34666340	0.20	0.57	0.29	0.18	0.42	0.26
lo_34669195	0.15	0.60	0.24	0.22	0.40	0.28
lo_34667655	0.20	1.00	0.33	0.03	0.25	0.06
lo_34668617	0.15	0.37	0.21	0.40	0.50	0.44
lo_34664558	0.45	0.69	0.54	0.33	0.38	0.35
lo_34667229	0.45	1.00	0.62	0.00	0.00	NaN
lo_34666923	0.45	0.75	0.56	0.10	0.25	0.14
lo_34667186	0.15	0.27	0.19	0.14	0.18	0.16
lo_34669900	0.15	0.42	0.22	0.22	0.71	0.34
lo_34665440	0.20	0.66	0.30	0.26	0.66	0.38
lo_34666277	0.10	0.66	0.17	0.15	0.66	0.25
lo_34666092	0.50	1.00	0.66	0.00	0.00	NaN
lo_34668948	0.20	0.57	0.29	0.16	0.71	0.26
lo_34666503	0.30	1.00	0.46	0.07	0.16	0.10
lo_34664521	0.00	0.50	0.09	0.16	1.00	0.28
lo_34666531	0.40	0.66	0.50	0.16	0.41	0.23
lo_34667002	0.70	0.73	0.71	1.00	0.36	0.52
lo_34668374	0.20	0.57	0.29	0.25	0.42	0.31

Tabla 5.5: Valores de precisión, cobertura y medida-F para ADEGA y RelFinder en la anotación de 40 metadatos de objetos de aprendizaje de Universia. *P*, *R*, and *F* stands for precision, recall, and f-score, respectively

Identificador LO	Nodos	Categorías	Identificador LO	Nodos	Categorías
21073	67	52	21133	56	38
21093	72	55	21134	34	23
21094	40	26	21135	38	26
21095	46	18	21136	24	11
21096	15	9	21137	79	58
21113	28	18	21138	197	55
21114	66	52	21139	13	6
21115	3	2	21140	70	41
21116	45	23	21141	8	6
21117	9	4	21142	40	22
21118	40	25	21143	110	80
21119	62	41	21144	62	30
21120	27	13	21145	53	27
21131	14	8	21146	49	33
21132	33	26	21147	70	33

Tabla 5.6: Resumen del número de nodos y categorías recuperadas para alguno de los LOs durante el proceso de filtrado.

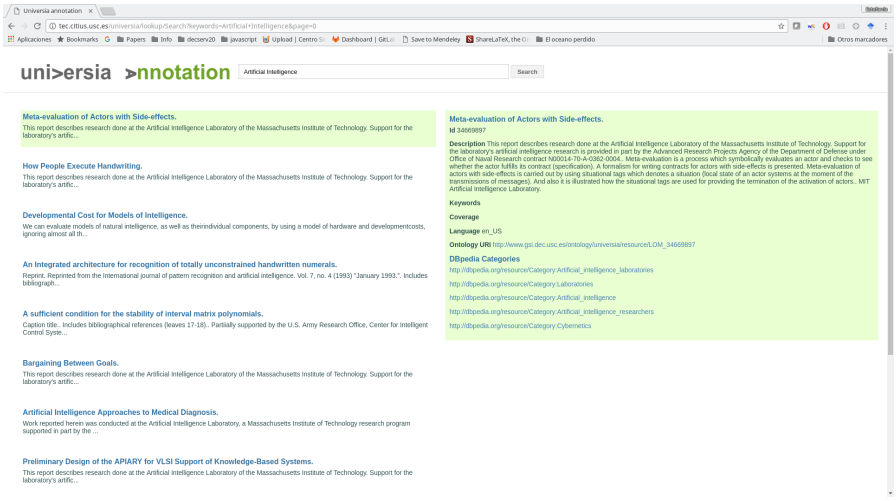

Figura 5.4: Servicio de búsqueda sobre el repositorio semantizado de Universia

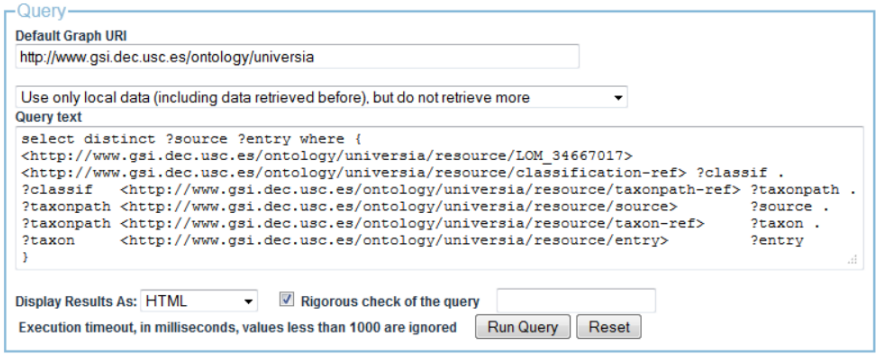

Figura 5.5: Consulta SPARQL al punto de acceso del repositorio semantizado de Universia

source	entry
DBpedia	http://dbpedia.org/resource/Category:Asia
DBpedia	http://dbpedia.org/resource/Category:Isolating_languages
DBpedia	http://dbpedia.org/resource/Category:Chinese_language
DBpedia	http://dbpedia.org/resource/Category:Languages_of_Asia
DBpedia	http://dbpedia.org/resource/Category:Language
DBpedia	http://dbpedia.org/resource/Category:Chinese_literature
DBpedia	http://dbpedia.org/resource/Category:Languages_of_China
DBpedia	http://dbpedia.org/resource/Category:Central_Asian_studies
DBpedia	http://dbpedia.org/resource/Category:Asian_American_writers
DBpedia	http://dbpedia.org/resource/Category:Asian_Americans
DBpedia	http://dbpedia.org/resource/Category:Hiking_trails_in_Asia
DBpedia	http://dbpedia.org/resource/Category:Americana
DBpedia	http://dbpedia.org/resource/Category:Asian_studies
DBpedia	http://dbpedia.org/resource/Category:Southeast_Asian_studies
DBpedia	http://dbpedia.org/resource/Category:Chinese_language_by_country

Figura 5.6: Resultado de la consulta SPARQL al repositorio semantizado de Universia

5.7. Conclusiones

En este capítulo, presentamos un caso de aplicación que demuestra la validez de nuestra aproximación de anotación en el ámbito de la categorización de objetos de aprendizaje, concretamente los objetos de la biblioteca digital de Universia. Esta propuesta mejora los resultados obtenidos con Relfinder, la única aproximación similar en el estado del arte en el momento en que se llevaron a cabo estas pruebas. Nuestra aproximación soporta la selección automática de las categorías más adecuadas, ordenadas de acuerdo a su relevancia y relación con el LO; con el añadido de que es una aproximación que trabaja con múltiples dominios, buscando una solución en más de 700.000 categorías de la DBpedia. Además esta propuesta puede extenderse con más conjuntos de categorías, incluyendo otros repositorios de datos enlazados, ya que el algoritmo explora el grafo de la ontología independientemente de sus relaciones y estructura; de hecho, en este caso sólo depende del vocabulario SKOS.

Finalmente, el repositorio de Universia ha sido transformado en un repositorio semántico siguiendo los principios de los datos enlazados, donde se puede acceder a los metadatos de cada objeto a través de una URI realizando una consulta al punto de acceso SPARQL o usando el buscador de palabras clave.

Conclusiones

La principal contribución de esta tesis ha sido el desarrollo de ADEGA, un sistema de anotación semántica que usa la DBpedia como fuente de datos enlazados para identificar, etiquetar y clasificar los términos relevantes de un documento de entrada. La DBpedia es un repositorio genérico (en inglés, *cross domain*) que ofrece cobertura a una gran cantidad de dominios, haciendo de ADEGA un anotador semántico de propósito general. Además, ADEGA también permite utilizar otros repositorios de datos enlazados, dado que la exploración del repositorio se realiza de forma independiente a su estructura o forma de representar el conocimiento. Para llevar a cabo el desarrollo, se ha seguido un esquema modular que permite identificar, por una parte, los términos relevantes de un documento y, por otra, anotar dichos términos con grafos extraídos de los repositorios de datos enlazados. En este sentido, las principales contribuciones del sistema propuesto son las siguientes:

- Se ha desarrollado un módulo de extracción de términos relevantes asociados a un documento, utilizando para ello técnicas de procesamiento de lenguaje natural para obtener los términos y calcular sus relevancias asociadas. Este objetivo inicialmente se consiguió mediante la generación de un conjunto de expresiones regulares basadas en construcciones gramaticales, con un análisis posterior de frecuencia, morfología y similitud. Este sistema tenía dos problemas principales: (*i*) la construcción del conjunto de reglas era dependiente del idioma, lo que afectaba negativamente a la forma de expresar las reglas, además de caer en falsos positivos o ignorar ciertas expresiones que podían estar mal escritas o expresadas en una forma diferente no recogida en las reglas; y (*ii*) podíamos detectar ciertas expresiones o términos que no estuvieran contenidos en el repositorio de datos enlazados que se usa como referencia, por lo que no sería posible anotar dicho término. Estos problemas, sumados a la necesidad de preprocesar la información de forma eficiente para mejorar el rendimiento, nos llevó a trabajar con

un algoritmo de *N-gramas* en la detección de los términos. El algoritmo hace uso de un índice de términos pre-procesados de la DBpedia, de forma que identifica los términos del repositorio en el documento independientemente de reglas gramaticales específicas del idioma. Utilizando frecuencias de aparición, frecuencias inversas, sinónimos y morfología de los términos, calculamos la relevancia asociada a ellos.

- Se ha desarrollado un módulo de desambiguación de conceptos para seleccionar, a partir de un conjunto de conceptos candidatos, aquel que se ajuste mejor al término del documento que se intenta anotar, obteniendo un único concepto con el que se representa. En una primera aproximación se obtuvo el conjunto de entidades candidatas para un término utilizando un servicio de consulta SPARQL de la DBpedia, el cual es independiente del sistema ADEGA. El principal problema de usar este servicio era doble: por una parte, no se tiene control sobre el tiempo de respuesta y, por otro, la forma de seleccionar el mejor candidato recae en el modo en el que se realiza la búsqueda SPARQL (equiparación sintáctica). Como alternativa, en la segunda versión de ADEGA, construimos nuestro propio servicio de búsqueda de entidades mediante la indexación de las etiquetas textuales de la DBpedia, de forma que podemos mejorar sustancialmente la selección de la entidad que está vinculada a cada término relevante y que constituye el nodo raíz del grafo solución para dicho término.

  Para desambiguar una entidad, planteamos un proceso de selección de la entidad correcta basado en el cálculo de relevancia asociada, entendiendo que esta relevancia viene dada por las relaciones de la entidad hacia otras entidades. Por lo tanto, se propone un algoritmo que explora el grafo de conceptos a partir de la entidad candidata y calcula su relevancia mediante el pesado de las relaciones y el resto de conceptos. De esta forma, se considera que la entidad que tiene una mayor relevancia es la entidad correcta. Este objetivo está íntimamente relacionado con nuestra principal contribución: el desarrollo de un algoritmo que nos permite explorar el grafo del repositorio a partir de un concepto inicial, filtrando aquella información que no está relacionada entre sí y pesando el contenido en relación a los términos identificados inicialmente. De hecho, una de las principales conclusiones de la tesis es que la mejora en el algoritmo de anotación semántica está muy condicionada por una adecuada selección de los términos relevantes del documento.

- Se ha desarrollado un algoritmo de exploración y filtrado del repositorio de datos en-

lazados con el que se obtienen los grafos que anotan los términos relevantes del documento. Inicialmente, se propuso un algoritmo DFS que exploraba de forma exhaustiva, y hasta un nivel de exploración parametrizable, el grafo de la DBpedia, pesando cada nodo y descartándolo según un umbral establecido. Esta aproximación tenía un grave problema de rendimiento: el 80 % de los nodos que explorábamos eran descartados, afectando negativamente en el rendimiento del algoritmo. Además, el 90 % de los nodos procesados eran nodos de texto plano que requerían la aplicación de técnicas de procesamiento de lenguaje natural para analizarlos, técnicas costosas en tiempo y rendimiento. Para resolver este problema, se propuso una paralelización del algoritmo utilizando un paradigma de computación distribuida que incluía el uso de recursos computacionales heterogéneos. Sin embargo, esta solución, aunque reduce considerablemente el tiempo de computación del algoritmo, tiene dos problemas: por una parte, no siempre es posible disponer de una infraestructura de computación de altas prestaciones para resolver los problemas de anotación y, por otra parte, la reducción del tiempo de procesado no es lo suficientemente elevada como para aplicar el algoritmo en ciertos dominios como, por ejemplo, las redes sociales.

Para solventar este problema se propuso un nuevo sistema de exploración del grafo que parte de información pre-procesada de la DBpedia para disminuir el tiempo de análisis de los datos textuales. Además, se enfocó el problema de la exploración del grafo como un problema de búsqueda de caminos entre nodos. De esta forma, no se parte de un conjunto de nodos para realizar exploración, sino que se introduce un conjunto de nodos finales temáticamente relacionados con los de partida, lo que permite recortar significativamente el espacio de búsqueda. Mediante la aplicación de un algoritmo DFS bidireccional se construyen los caminos entre los nodos y se pesa el grafo obtenido utilizando dos medidas: una de similitud entre el texto del nodo con los términos iniciales; y una medida de centralidad de grado, que nos permite evaluar la topología del grafo creado.

Finalmente, es importante reseñar que la salida del algoritmo ADEGA no es sólo una anotación de los términos relevantes del texto mediante el enlazado a una entidad de un repositorio, sino también un grafo de conceptos asociado a esta entidad que contiene información relacionada al texto de partida, haciendo que el contenido se vea enriquecido. Este grafo nos permite explotar la semántica y las relaciones entre las entidades; con el objetivo de poder utilizar esta información para mejorar procesos de búsqueda y

recomendación.

Habiendo desarrollado todos los módulos, se realizó una comparativa de ADEGA con el resto de los sistemas de anotación disponibles en el estado del arte, usando para ello los conjuntos de datos públicamente disponibles que están aceptados en la comunidad investigadora. Como resultado de esta comparativa, se concluye que, por lo general, ADEGA tiene un mejor rendimiento que los algoritmos del estado del arte. Más concretamente, obtiene el mejor F1 promedio en las tareas C2W y D2W, con un valor de 56,73 % y 80,73 %, respectivamente. El segundo mejor resultado lo obtiene *TagMe2*, con 52,78 % en C2W y 72,72 % en D2W; y el tercero, *DBpediaSpotlight*, con 36,07 % y 70,85 %, respectivamente. Cabe destacar los resultados de ADEGA en el ámbito de D2W, lo que indica el buen comportamiento del algoritmo de exploración y filtrado, que permite focalizar la búsqueda en aquellas entidades que son realmente relevantes al tratarse de un problema de enlazado de términos a conceptos. Debido a la ausencia de algoritmos y conjuntos de datos que anoten términos con grafos semánticos, no hemos podido evaluar la contribución de nuestro sistema en relación al enriquecimiento de información contra otras aproximaciones. No obstante, sí hemos tenido la oportunidad de evaluar *ADEGA* en el ámbito del enriquecimiento de información educativa debido a la construcción de un conjunto de datos específico de la mano de un conjunto de expertos.

Trabajo futuro

Si bien los resultados de ADEGA en relación a la calidad de la anotación semántica superan a las propuestas actuales del estado del arte, mostrando también un buen comportamiento en cuanto a su rendimiento y capacidad para procesar grandes conjuntos de datos, se han identificado algunas mejoras que se podrían introducir a los diferentes módulos de ADEGA y que, a su vez, mejorarían su comportamiento global. Estas mejoras están ligadas a la identificación de términos relevantes, la exploración y filtrado de los grafos semánticos, y a las medidas de similitud entre términos. Más concretamente, las mejoras que se proponen para el trabajo futuro son las siguientes:

- Introducción de nuevas técnicas para el cálculo de la similitud entre términos. El uso de técnicas de similitud entre términos basadas en contexto y en la coocurrencia de aparición de dichos términos en corpus extensos podría mejorar el cálculo de la relevancia de los términos del contexto y de las relaciones textuales de las entidades. Esta mejora

vendría dada por la introducción de la distancia entre términos dentro de la fórmula para el cálculo de su relevancia (sección 4.6). Entre estas técnicas destacamos *Word2vec* [69] en la que se usaría la Wikipedia como el corpus en el que se explorarían las relaciones entre los términos.

- Indexación de caminos precalculados del grafo de la DBpedia a altos niveles de profundidad, para mejorar el rendimiento del algoritmo de exploración y filtrado. En la versión actual de ADEGA, se almacenan las relaciones de primer nivel de la DBpedia, de modo que haciendo uso del algoritmo de exploración bidireccional presentado en el sección 4.4.1, se divide por dos el tiempo necesario para construir los caminos entre los nodos. Sin embargo, cuando el número de términos relevantes del contexto es muy elevado, el tiempo requerido para la anotación del documento sigue siendo considerable. Para tratar este problema sería conveniente aumentar la profundidad en el almacenamiento de los caminos entre las entidades (o nodos) hasta, idealmente, llegar a almacenar todos los caminos de conexión entre todos los nodos del grafo semántico. De esta forma, se evitaría la fase de exploración de los grafos, ya que determinar los caminos de conexión entre dos nodos se reduciría a una consulta en la base de datos que contiene todos los caminos posibles.

- Configuración del algoritmo basado en la caracterización del documento de entrada. Una de las conclusiones extraídas de la comparación entre los algoritmos de anotación semántica del estado del arte es que la eficacia de las técnicas para el cálculo de la relevancia de las entidades depende de las características del documento de entrada y de la cohesión semántica entre sus términos relevantes. Por ejemplo, en algunos conjuntos de documentos la medida de centralidad de grado usada por ADEGA se comporta mejor que los algoritmos de PageRank, HITS o camino aleatorio; mientras que en otros documentos se demuestra lo contrario. Así, la caracterización de los documentos para seleccionar en tiempo de ejecución las mejores medidas para el cálculo de la relevancia de entidades permitiría adaptar ADEGA a dichos documentos, lo cual mejoraría la calidad de los resultados a expensas de una pequeña penalización del rendimiento.

Además de estas mejoras en los módulos de ADEGA, como trabajo futuro también se plantea ampliar el sistema con repositorios semánticos distintos de la DBpedia (como, por ejemplo, Freebase [10]) y con grandes ontologías en dominios de propósito específico. En este sentido, es especialmente interesante el uso de estándares como *MeSH* [1] y *SNOMED-*

CT [2] en el ámbito médico, no solo para la anotación de informes o artículos científicos, sino también para comentarios generados por usuarios en redes sociales.

Bibliografía

[1] Medical subject headings (mesh). *https://www.nlm.nih.gov/mesh/*, 1960.

[2] Systematized nomenclature of medicine – clinical terms. *http://www.snomed.org/*, 2002.

[3] Biblioteca universia recursos aprendizaje. *http://biblioteca.universia.net/*, 2012.

[4] Hend S Al-Khalifa and Hugh C Davis. Fasta: A folksonomy-based automatic metadata generator. In *European Conference on Technology Enhanced Learning*, pages 414–419. Springer, 2007.

[5] Ricardo R. Amorim, Manuel Lama, Eduardo Sánchez, Adolfo Riera, and Xosé A. Vila. A learning design ontology based on the ims specification. *Educational Technology & Society*, 9(1):38–57, 2006.

[6] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[7] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The Semantic Web. *Scientific american*, 284(5):28–37, 2001.

[8] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.

[9] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web

of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165, 2009.

[10] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[11] Mihaela M Brut, Florence Sedes, and Stefan Daniel Dumitrescu. A semantic-oriented approach for organizing and developing annotation for e-learning. *IEEE Transactions on Learning Technologies*, 4(3):239–248, 2011.

[12] Razvan C. Bunescu and Marius Pasca. Using Encyclopedic Knowledge for Named entity Disambiguation. *EACL*, 6:9–16, 2006.

[13] Manuel Caeiro-Rodriguez, Roberto Perez-Rodriguez, Javier Garcia-Alonso, Mario Manso-Vazquez, and Martin Llamas-Nistal. Area: A social curation platform for open educational resources and lesson plans. In *2013 IEEE Frontiers in Education Conference*, pages 795–801. IEEE, 2013.

[14] Manuel Caeiro-Rodríguez, Juan M. Santos-Gago, Manuel Lama, and Martin Llamas-Nistal. A keyword recommendation experiment to support information organization and folksonomies in edu-area. *IEEE Revista Iberoamericana de Tecnologías del Aprendizaje*, 10(2):60–68, 2015.

[15] David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. Erd'14: entity recognition and disambiguation challenge. In *ACM SIGIR Forum*, volume 48, pages 63–77. ACM, 2014.

[16] Nicholas Carriero and David Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, 1989.

[17] Jesus Soto Carrion, Elisa Garcia Gordo, and Salvador Sanchez-Alonso. Semantic learning object repositories. *International Journal of Continuing Engineering Education and Life Long Learning*, 17(6):432–446, 2007.

[18] Danilo S. Carvalho, André Danilo, and Joao C.P. da Silva. Graphia: Extracting Contextual Relation Graphs from Text. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 236–241. Springer, 2013.

[19] Ping-I Chen and Shi-Jen Lin. Word adhoc network: using google core distance to extract the most relevant information. *Knowledge-Based Systems*, 24(3):393–405, 2011.

[20] Rudi L. Cilibrasi and Paul MB. Vitanyi. The google similarity distance. *IEEE Transactions on knowledge and data engineering*, 19(3):370–383, 2007.

[21] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, volume 3, pages 73–78, 2003.

[22] Thomas F. Coleman and Jorge J. Moré. Estimation of sparse jacobian matrices and graph coloring blems. *SIAM journal on Numerical Analysis*, 20(1):187–209, 1983.

[23] IEEE Learning Technology Standards Committee et al. Draft standard for learning object metadata. *Accessed July*, 14, 2002.

[24] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 249–260. ACM, 2013.

[25] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. *EMNLP-CoNLL*, 7:708–716, 2007.

[26] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. A framework and graphical development environment for robust nlp tools and applications. In *ACL*, pages 168–175, 2002.

[27] Juan Manuel Dodero, Paloma Díaz, Ignacio Aedo, and Antonio Sarasa Cabezuelo. Integrating ontologies into the collaborative authoring of learning objects. *Journal of Universal Computer Science*, 11(9):1568–1578, 2005.

[28] Milan Dojchinovski and Tomáš Kliegr. Entityclassifier. eu: Real-time classification of entities in text with Wikipedia. In *Machine Learning and Knowledge Discovery in Databases*, pages 654–658. Springer, 2013.

[29] Javier Fabra, Sergio Hernández, Pedro Álvarez, and Joaquín Ezpeleta. A framework for the flexible deployment of scientific workflows in grid environments. In *Cloud Compiting 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 43–50, 2012.

[30] Javier Fabra, Sergio Hernández, Joaquín Ezpeleta, and Pedro Álvarez. Solving the interoperability problem by means of a bus. *Journal of Grid Computing*, 12(1):41–65, 2014.

[31] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[32] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628, 2010.

[33] Besnik Fetahu, Stefan Dietze, Bernardo Pereira Nunes, Davide Taibi, and Marco Antonio Casanova. Generating structured Profiles of Linked Data Graphs. In *International Semantic Web Conference (Posters & Demos)*, pages 113–116, 2013.

[34] Anna Lisa Gentile, Ziqi Zhang, Lei Xia, and José Iria. Graph-based semantic relatedness for named entity disambiguation. In *Proceedings of International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES*, page 13. S3T, 2009.

[35] Adriana Gewerc, Lourdes Montero, and Manuel Lama. Collaboration and social networking in higher education. *Comunicar*, 21(42):55–63, 2014.

[36] Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *COLING*, volume 96, pages 466–471, 1996.

[37] Ben Hachey, Will Radford, and James R. Curran. Graph-based named entity linking with Wikipedia. In *Web Information System Engineering–WISE 2011*, pages 213–226. Springer, 2011.

[38] Sherzod Hakimov, Salih Atilay Oto, and Erdogan Dogdu. Named entity recognition and disambiguation using linked data and graph-based centrality scoring. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, page 4. ACM, 2012.

[39] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM, 2011.

[40] Xianpei Han and Jun Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. *Proceedings of the 18th ACM conference on Information and Knowledge Management*, pages 215–224, 2009.

[41] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[42] Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. Relfinder: Revealing relationships in rdf knowledge bases. In *Semantic Multimedia*, pages 182–187. Springer, 2009.

[43] Diane Hillmann. Using Dublin Core, Dublin Core Qualifiers. 2003.

[44] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[45] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R. Pocock, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic acids research*, 34(Supplement 2):W729–W732, 2006.

[46] Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. An Eigenvalue-Based Measure for Word-Sense Disambiguation. In *The Florida Artificial Intelligence Research Society, FLAIRS Conference*, 2012.

[47] Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the sixth ACM international conference on Web Search and Data Mining*, pages 465–474. ACM, 2013.

[48] Ioana Hulpuş, Conor Hayes, Marcel Karnstedt, Derek Greene, and Marek Jozwowicz. Kanopy: Analysing the semantic network around document topics. In *Machine Learning and Knowledge Discovery in Databases*, pages 677–680. Springer, 2013.

[49] Dublin Core Metadata Initiative et al. Dcmi metadata terms. *http://www.dublincore.org/documents/dcmi-terms/*, 2016-06-14.

[50] Jelena Jovanovic, Dragan Gasevic, and Vladan Devedzic. Ontology-based automatic annotation of learning content. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2(2):91–119, 2006.

[51] José Kahan, M-R Koivunen, Eric Prud'Hommeaux, and Ralph R Swick. Annotea: an open rdf infrastructure for shared web annotations. *Computer Networks*, 39(5):589–608, 2002.

[52] Gjergji Kasneci, Shady Elbassuoni, and Gerhard Weikum. Ming: mining informative entity relationship subgraphs. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1653–1656. ACM, 2009.

[53] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49–79, 2004.

[54] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[55] Graham Klyne and Jeremy J. Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

[56] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM, 2009.

[57] Olaf Kummer, Frank Wienberg, Michael Duvigneau, Jörn Schumacher, Michael Köhler, Daniel Moldt, Heiko Rölke, and Rüdiger Valk. An extensible editor and simulation engine for petri nets: Renew. In *International Conference on Application and Theory of Petri Nets*, pages 484–493. Springer, 2004.

[58] Erwin Laure, S. Fisher, A. Frohner, Claudio Grandi, Peter Kunszt, Ales Krenek, Ole Mulmo, Fabrizio Pacini, Francesco Prelz, John White, et al. Programming the grid with glite. *Computational methods in science and technology*, 12(1):33–45, 2006.

[59] Chin Yang Lee. An algorithm for path connections and its applications. *IRE transactions on electronic computers*, pages 346–365, 1961.

[60] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, van Patrick Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[61] László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2:1–46, 1993.

[62] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[63] Christopher D. Manning, Prabhakar Raghavan, and HinrichSchütze. *Introduction to information retrieval*, volume 1. Cambridge: Cambridge university press, 2008.

[64] David Massart. A "simple query interface" adapter for the discovery and exchange of learning resources. *International Journal on E-Learning*, 5(1):151–159, 2006.

[65] Rory McGreal. A typology of learning object repositories. In *Handbook on information technologies for education and training*, pages 5–28. Springer, 2008.

[66] Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 563–572. ACM, 2012.

[67] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems (I- SEMANTICS 2011)*, pages 1–8. ACM, 2011.

[68] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. *Proceedings of the 16th ACM conference on Information and Knowledge Management*, pages 233–242, 2007.

[69] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[70] Alistair Miles and Sean Bechhofer. Skos simple knowledge organization system reference. *https://www.w3.org/TR/skos-reference/*, 2009.

[71] David Milne and Ian H. Witten. Learning to link with wikipedia. *Proceedings of the 17th ACM conference on Information and Knowledge Management*, pages 509–518, 2008.

[72] Roberto Mirizzi, Azzurra Ragone, Tommaso Di Noia, and Eugenio Di Sciascio. Semantic tag cloud generation via DBpedia. In *E-Commerce and Web Technologies*, pages 36–48. Springer, 2010.

[73] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity Linking meets Word Sense Disambiguation: A Unified Approach. *Transactions of the Association for Computational Linguistics*, 2, 2014.

[74] O. Muñoz-García, Andrés García-Silva, Óscar Corcho, M. Higuera Hernández, and Carlos Navarro. Identifying topics in social media posts using DBpedia. In *Proceedings of the Networked and Electronic Media Summit (NEM summit 2011)*, 2011.

[75] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[76] Roberto Navigli and Mirella Lapata. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):678–692, 2010.

[77] Roberto Navigli and Simone Paolo Ponzetto. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[78] Saša Nešić, Dragan Gašević, Mehdi Jazayeri, and Monica Landoni. A learning content authoring approach based on semantic technologies and social networking: An empirical study. *Supporting Organizations*, page 35, 2011.

[79] Allen Newell. The knowledge level. *Artificial intelligence*, 18(1):87–127, 1982.

[80] Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. AIDA-light: High-Throughput Named-Entity Disambiguation. In *Linked Data on the Web (LDOW2014)*, 2014.

[81] Katja Niemann and Martin Wolpers. Enabling the use of real world objects to improve learning. In *2010 10th IEEE International Conference on Advanced Learning Technologies*, pages 259–263. IEEE, 2010.

[82] Xavier Ochoa, Joris Klerkx, Bram Vandeputte, and Erik Duval. On the use of learning object metadata: The globe experience. In *Towards ubiquitous learning*, pages 271–284. Springer, 2011.

[83] Estefanıa Otero-Garcıa, Juan C. Vidal, Manuel Lama, Alberto Bugarın, and José E. Domenech. A context-based algorithm for annotating educational content with linked data. In *3rd Future Internet Symposium (FIS 2010), FIS-Workshop on Mining Future Internet (MIFI 2010)*, 2010.

[84] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[85] Claus Pahl and Edmond Holohan. Applications of semantic web technology to support learning content development. *Interdisciplinary Journal of E-Learning and Learning Objects*, 5, 2009.

[86] Eric Prud'Hommeaux and Andy Seaborne. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[87] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009.

[88] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics, 2011.

[89] Thomson Reuters. Opencalais. *http://www.opencalais.com/*.

[90] Giuseppe Rizzo, Raphaël Troncy, Sebastian Hellmann, and Martin Bruemmer. Nerd meets nif: Lifting nlp extraction results to the linked data cloud. *LDOW*, 937, 2012.

[91] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[92] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[93] Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-based graph document modeling. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 543–552. ACM, 2014.

[94] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, 2015.

[95] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, pages 449–458. ACM, 2012.

[96] Frank Spitzer. *Principles of random walk*. Springer, 1964.

[97] Fabian M. Suchanek and Gjergji Kasneciand Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[98] Martin Svensson, Arianit Kurti, and Marcelo Milrad. Enhancing emerging learning objects with contextual metadata using the linked data approach. In *Wireless, Mobile and Ubiquitous Technologies in Education (WMUTE), 2010 6th IEEE International Conference on*, pages 50–56. IEEE, 2010.

[99] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[100] Stefaan Ternier, Katrien Verbert, Gonzalo Parra, Bram Vandeputte, Joris Klerkx, Erik Duval, V. Ordoez, and Xavier Ochoa. The ariadne infrastructure for managing and storing metadata. *IEEE Internet Computing*, 13(4):18–25, 2009.

[101] Andraž Tori and Tomaž Šolc. Zemanta service. *http://www.zemanta.com/*.

[102] Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very*

large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13, pages 63–70. Association for Computational Linguistics, 2000.

[103] Scientific United Nations Educational and Cultural Organization (UNESCO). Proposed international standard nomenclature for fields of science and tecnology. *(UNESCO/NS/ROU/257)*, 1988.

[104] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, Sandro Athaide Coelho, Sören Auer, and Andreas Both. AGDISTIS-Graph-Based Disambiguation of Named Entities using Linked Data. In *International Semantic Web Conference–ISWC 2014*, pages 457–471. Springer, 2014.

[105] Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, et al. Gerbil: General entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1133–1143. ACM, 2015.

[106] Andrea Varga, Amparo Elizabeth Cano Basave, Matthew Rowe, Fabio Ciravegna, and Yula He. Linked Knowledge Sources for Topic Classification of Microposts: A Semantic graph-based approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2014.

[107] Juan Carlos Vidal, Manuel Lama, Estefanía Otero-García, and Alberto Bugarín. An evolutionary approach for learning the weight of relations in linked data. In *11th International Conference on Intelligent Systems Design and Applications (ISDA 2011)*, pages 1002–1007. IEEE Computer Society, 2011.

[108] David A. Wiley. *Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy*. 2003.

[109] William E. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, U.S. Census Bureau*. Citeseer, 1999.

Índice de figuras

Índice de tablas