Obtención de funciones óptimas para el matching de ontologías usando algoritmos heurísticos

Jorge Martinez-Gil

Universidad de Málaga, Departmento de Lenguajes y Ciencias de la Computación
Boulevard Louis Pasteur s/n 29071 Málaga (España)
{jorgemar,jfam}@lcc.uma.es
http://www.lcc.uma.es

Abstract. Actualmente existen multitud de técnicas y herramientas para resolver el problema del matching de ontologías. No obstante, la compleja naturaleza del problema hace que las soluciones existentes no sean completamente satisfactorias. Este trabajo intenta arrojar luz sobre una manera más flexible para emparejar ontologías: Ontology Meta-Matching. En este sentido, pensamos que un estudio exaustivo del problema y una revisión de las soluciones más notables puede ayudarnos a desarrollar sistemas más precisos y dinámicos que permitan seleccionar de manera automática los algoritmos y los parámetros asociados necesarios para resolver problemas de interoperabilidad entre ontologías en la Web Semántica.

Key words: meta-matching, ontologías, web semántica

1 Introducción

La Web Semántica es un paradigma en el que la semántica de la información contenida en la World Wide Web (WWW) está definida haciendo posible que se comprenda y se responda de manera automática a las peticiones de las personas y las máquinas que desean usar los recursos web. Por lo tanto, la mayoría de los autores consideran este paradigma como un medio universal para el intercambio de datos, información y conocimiento [1].

En relación al conocimiento, es muy importante la noción de ontología como la forma de representación sobre un universo de discurso o una parte de él. El Ontology Matching (en castellano emparejamiento de ontologías) es un aspecto clave para que el intercambio de conocimiento en esta extensión de la Web sea real, pues permite a las organizaciones modelar su propio conocimiento sin tener que ajustarse a un estándar específico. De hecho, hay dos buenas razones por las que la mayoría de las organizaciones no están interesadas en trabajar con un estándar para modelar el conocimiento con el que trabajan: (a) Puede ser muy difícil o costoso para muchas organizaciones alcanzar un acuerdo sobre un estándar para modelar su propio conocimiento y (b) a menudo estos estándares no se ajustan a las necesidades específicas de todos los participantes en el proceso de estandarización.

Por tanto, el antiguo problema del emparejamiento de esquemas ha evolucionado a uno análogo, aunque un poco más complejo, pues ahora manejamos conocimiento. La tarea de encontrar correspondencias entre ontologías se llama Ontology Matching y la salida de esta tarea se conoce con el nombre Alineamiento de Ontologías [2]. De hecho, obtener alineamientos satisfactorios es un aspecto clave para campos como:

- Integración semántica [3]. Es el proceso de combinar metadatos localizados en diferentes fuentes de manera que se ofrece al usuario una vista unificada de esos datos. Este tipo de integración debería hacerse de manera automática, debido a que la integración manual no es viable, al menos, para grandes volúmenes de información [4].
- Ontology mapping [5]. Se usa para consultar simultáneamente diferentes ontologías. Un mapping es una función entre ontologías. Las ontologías originales no cambian, sino los axiomas de correspondencia entre conceptos, relaciones o instancias. Que están separados de dichas ontologías. Un caso de uso típico es el de una consulta sobre una ontología que se ha evolucionado. Las respuestas deben calcularse de nuevo. Mientras que el alineamiento sólo identifica relaciones entre ontologías, las funciones de mapping se centran en la representación y en la ejecución de las relaciones para una determinada tarea.
- Servicios Web, donde se tiende al descubrimiento y la composición de Servicios Web Semánticos [6] sin supervisión alguna. En un principio, el alineamiento de SWS estaba basado en el matching exacto de parámetros, pero a día de hoy, los investigadores tratan problemas de heterogeneidad más complicados[1].
- Recuperación de la información basada en la similitud [7]. La similitud semántica juega un papel importante en la recuperación de la información pues ofrece medios para mejorar la precisión y la cobertura. Se usa en varios dominios de aplicación que abarcan desde la comparación de productos a la selección de candidatos.

Por otra parte, aunque el emparejamiento semántico es, quizás, la forma más apropiada para alinear ontologías, tiene la desventaja de que encontrar buenas funciones de similitud es dependiente de los datos, del contexto e incluso de los usuarios y necesita recalcularse cada vez que hay nuevos datos [8]. Además, tratar con problemas relativos al lenguaje natural suele inducir una tasa de error significativa, por lo que los investigadores se afanan en conseguir funciones de similitud personalizadas (CSF) para obtener el alineamiento que mejor se ajusta a cada situación.

Por otra parte, las funciones para calcular alineamientos pueden dividirse en medidas de similitud y distancias.

- Un medida de similitud es una función que asocia un valor numérico a un par de objetos, con la idea de que un valor más alto indica una similitud mayor.

[1] http://insel.flp.cs.tu-berlin.de/wsc06/

– Un distancia de similitud es un función que asocia un valor numérico no negativo con un par de objetos con la idea de que una distancia mayor significa mayor similitud. Las distancias satisfacen los axiomas de una métrica.

Las leyes matemáticas usadas para describir el comportamiento en un dominio no son siempre apropiadas en otros dominios, pues hay objeciones psicológicas para los axiomas usados en la definición de una distancia. Por ejemplo, una distancia nos dará siempre la misma distancia entre a y b que entre b y a, pero en la práctica es más preciso decir que un niño se parece a su padre que un padre se parece a su hijo. Las medidas de similitud, sin embargo, nos dan una idea acerca de la probabilidad de que dos objetos son el mismo, pero sin caer en las objeciones psicológicas de una métrica. Por lo que desde nuestro punto de vista, trabajar con medidas de similitud es más apropiado para detectar correspondencias entre entidades diferentes que pertenecen a un mismo dominio. En este sentido, el Ontology Meta-Matching podría considerarse como una técnica que selecciona de manera automática las medidas de similitud apropiadas y sus pesos asociados en caso de que las medidas necesiten componerse. Las principales contribuciones de este trabajo son:

– Una introducción al problema del Ontology Meta-Matching.
– Una medida de similitud estructural para alinear ontologías (SIFO).
– Una medida de similitud lingüísitica que usa motores de búsqueda en Internet para alinear ontologías.
– Una medida de similitud estadística que usa análisis textual para alinear ontologías.
– Un algoritmo voraz para solver el problema del Meta-Matching de manera automática y eficiente (MaSiMe).
– Un algoritmo genético para optimizar las soluciones del problema (GOAL).
– Un evaluación empírica de los algoritmos propuestos y una discusión sobre su conveniencia.
– Un estudio exaustivo de las futuras líneas de investigación en este campo

El resto de este trabajo está organizado de la manera siguiente. La Sección 2 describe el problema del Ontology Meta-Matching. La Sección 3 presenta las definiciones y propiedades que son necesarias para seguir nuestra propuesta. La Sección 4 describe el desarrollo de un algoritmo para calcular un medida de similitud estructural. La Sección 5 describe una medida de similitud lingüística que usa Internet como fuente de conocimiento. La Sección6 describe la implementación de un método estadístico para el alineamiento de ontologías. La Sección 7 describe un algoritmo voraz y una forma de implementación eficiente. La Sección 8 describe un algoritmo genético para optimizar las soluciones presentadas por el resto de algoritmos. La Sección 9 muestra los datos empíricos que hemos obtenidos a partir de varios experimentos, incluyendo los resultados obtenidos para un benchmark estándar. La Sección 10 incluye el trabajo relacionado, en el que comparamos nuestro trabajo con otras aproximaciones. Y finalmente, en la Sección 11 se presentan las conclusiones y las líneas de trabajo futuro.

2 Definición del problema

El proceso de alinear ontologías puede expresarse como una función f donde dados un par de ontologías o y o', un alineamiento de entrada A, un conjunto de parámetros p y un conjunto de recursos r, se devuelve un alineamiento A':

$$A' = f(o, o', A, p, r)$$

Donde A' es un conjunto de mappings. Un mapping[2] es una expresión que puede escribirse de la forma (id, e, e', n, R). Donde id es un identificador único del mapping, e y e' son entidades que pertenecen a distintas ontologías, R es la relación de correspondencia y n es un número real entre 0 y 1 que representa la probabilidad matemática de que R sea cierta [9]. Las entidades que pueden estar relacionadas son los conceptos, propiedades y reglas que componen las ontologías.

Sin embargo, la experiencia nos dice que encontrar f no es una tarea trivial. Como comentamos anteriormente, la heterogeneidad y la ambigüedad de las descripciones de los datos hace inviable encontrar los mejores mappings para pares de entidades dadas usando algoritmos de matching[3]. Por esta razón, es necesario componer estos matchers. La Figura 1 muestra un ejemplo de un alineamiento dependiente del usuario entre ontologías. Obsérvese que el alineamiento no es válido para todos los países del mundo.

Los matchers compuestos son agregaciones de algorimos de matching simples. Este tipo de matchers hacen uso de un amplio abanico de características de una ontologías (por ejemplo, nombres de elementos, tipos de datos, propiedades estructurales), características de las instancias así como fuentes externas de conocimiento como diccionarios y tesauros.

1. **Normalización de cadenas de texto.** Consiste en métodos que eliminan palabras innecesarias o símbolos de las etiquetas de las entidades antes de comparar las etiquetas. Además, puede usarse para detectar nombres en plural o para tener en cuenta prefijos o subfijos comunes así como otras características del lenguaje natural.
2. **Comparación de cadenas de texto.** La similitud de cadenas de texto es un método basado en la identificación de parecidos entre nombres de entidades. Por ejemplo, puede usarse para identificar conceptos idénticos en dos ontologías. El lector puede acudir a [10] para profundizar más en este tipo de algoritmos.
3. **Comparación de tipos de datos.** Estos métodos comparan el tipo de datos de los elementos de la ontología. Si dos entidades son equivalentes, es de esperar que tengan el mismo tipo de datos.

[2] No se deben confundir la denomicación de las tuplas de salida de un alineamiento a las que se suele denominar mapping, con la tarea de consultar ontologías de manera simultánea a la que también se denomina mapping

[3] Normalmente a los algoritmos de matching se les llama matchers

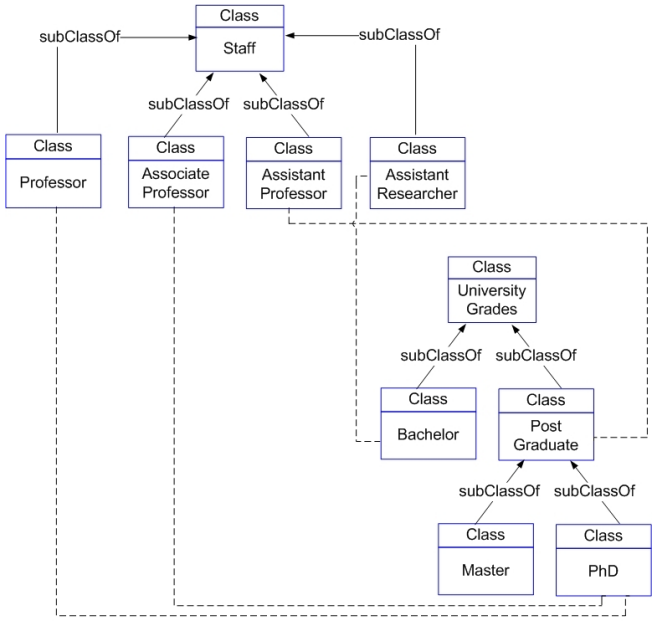


Fig. 1. Ejemplo de un alineamiento dependiente del usuario

4. **Métodos lingüísticos.** Consiste en el uso de recursos lingüñisticos tales como lexicones y tesauros para identificar posibles similitudes. El método lingüístico más popular es el que hace uso de WordNet [11] para identificar relaciones entre entidades.
5. **Análisis de herencia.** Estos tipos de métodos tienen en cuenta la herencia entre conceptos para identificar relaciones. El método más conocido es el del análisis *es-un* que intenta identificar subsumpciones entre conceptos.
6. **Análisis de datos.** Estos tipos de métodos están basados en la regla: Si dos conceptos tienen las mismas instancias, probablemente sean el mismo concepto. Algunas veces, es posible identificar el significado de una entidad de nivel superior mirando sus entidades de nivel inferior. Por ejemplo, si las instancias contiene una cadena de texto del tipo *años* probablemente pertenecen a un atributo llamado *edad*.
7. **Correspondencia de grafos.** Consiste en identificar estructuras de grafo parecidas en dos ontologías. Estos métodos usan algoritmos bien conocidos sobre grafos para hacerlo. La mayor parte de las veces, esto implica procesar y comparar caminos, nodos adyacentes además de nodos fuente y nodos sumidero.
8. **Análisis estadístico.** Consiste en la extracción de palabras claves y descripciones textuales para detectar el significado de unas entidades en relación a otras entidades.
9. **Análisis taxonómico.** Intenta identificar conceptos similares mirando sus conceptos relacionados. La principal idea es que dos conceptos que pertenecen

a distintas ontologías tienen cierto grado de probabilidad de ser iguales si tienen los mismos vecinos.

10. **Métodos semánticos.** Según [2], los algoritmos semánticos manejan las entradas en base a su interpretación semántica. Suponiendo que si dos entidades son la misma, entonces comparten la misma interpretación. Son por tanto métodos deductivos. Las propuestas más sobresalientes están basadas en la lógica de descripciones.

No obstante, elegir entre esta amplía variedad de algoritmos no es una tarea trivial. Primero, porque se desarrollan nuevos algoritmos constatemente y esta diversidad complica la elección del método más apropiado para una tarea dada. Segundo, porque un análisis empírico muestra que no hay ningún algoritmo de matching que destaque sobre el resto de manera independiente al modelo de datos y al dominio de aplicación [12]. De hecho, debido a la heterogeneidad y ambigüedad de las descripciones textuales, parece que no es posible decidir cuales son las funciones de mapping óptimas. Por esta razón, suelen usarse algoritmos de matching compuestos que equilibren varios aspectos a tener en cuenta a la hora de descubrir correspondencias.

La idea principal que subyace detrás de los algoritmos de matching compuesto es combinar los valores de similitud que han devuelto varias medidas de similitud para determinar las correspondencias entre los elementos de las ontologías. Las propuestas más destacadas en este sentido son: COMA [13], COMA++ [14], QuickMig [15], FOAM [16], iMAP [17], OntoBuilder [18], Cupid [19], CMC [20] y MAFRA [21], pero estas propuestas usan composiciones de algoritmos fijadas por un experto en el mejor de los casos. El Meta-Matching no usa pesos provenientes de un experto, sino que selecciona aquellos que podrían solucionar el problema conforme a un benchmark de entrenamiento, es decir, conforme a un conjunto lo suficientemente amplio y heterógeneo de ontologías que ha sido alineado previamente por un experto.

3 Preeliminares técnicos

Definición 1 (Medida de similitud). *Una medida de similitud sm es una función* $sm : \mu_1 \times \mu_2 \mapsto R$ *que asocia dos entidades* μ_1 *y* μ_2 *a un valor de similitud* $sc \in \Re$ *en el rango continuo [0, 1].* Está definición ha sido tomada y traducida de [12].

Un valor de similitud de 0 indica una desigualdad totla y un valor de 1 indica una completa igualdad entre μ_1 y μ_2.

Definición 2 (Medida de similitud parametrizable). *Una medida de similitud parametizable es aquella medida de similitud con aspectos que pueden definirse en función de parámetros* El Ejemplo 1 muestra una medida de similitud de este tipo.

Ejemplo 1 (Medida de similitud con pesos asociados). Sea $\boldsymbol{A}$ un conjunto de medidas de similitud, $\boldsymbol{w}$ un vector de pesos y sean O_1, O_2 dos ontologías de entrada, entonces podemos definir wsm como una medida de similitud con pesos asociados de la manera siguiente

$$wsm(O_1, O_2) = x \in [0,1] \in \Re \rightarrow \exists \langle \boldsymbol{A}, \boldsymbol{w} \rangle, x = max(\sum_{i=1}^{i=n} A_i \cdot w_i)$$
$$\textit{con la siguiente restricción } \sum_{i=1}^{i=n} w_i \leq \kappa$$

Pero desde el punto de vista de la ingeniería, esta función conduce a un problema de optimización a la hora de calcular el vector de pesos, porque el número de candidatos del espacio de soluciones (en este caso un intervalo real y continuo) es infinito. Por esta razón, una estrategia de fuerza bruta sería claramente ineficiente. Es necesario buscar mejores mecanismos computacionales que permitan que el problema de calcular medidas con pesos asociados se resuelva de manera más eficiente.

Definición 3 (Ontology Matching). *Una función de Ontology Matching om es una función* $om : O_1 \times O_2 \stackrel{sm}{\rightarrow} A$ *que asocia dos ontologías de entrada* O_1 *and* O_2 *a un alineamiento* A *usando una medida de similitud (o una medida de similitud parametrizable).*

Definición 4 (Alineamiento de ontologías). *Un alineamiento de ontologías oa es un conjunto* $\{t, MD\}$*. Donde* t *es un conjunto de tuplas de la forma* $\{(id, e, e', n, R)\}$*. Donde* id *es un identificador único,* e *y* e' *sone entidades que pertenecen a dos ontologías,* R *es la relación de correspondencia entre estas entidades y* n *es un número real entre 0 y 1 que representa la probabilidad matemática de que* R *sea cierta. Las entidades relacionadas pueden ser conceptos, roles o axiomas de las ontologías. Por otra parte, MD es un conjunto de metadatos relacionados con el proceso de matching recolectados con propósitos estadísticos.*

Definición 5 (Evaluación de un alineamiento). *Una evuluación de un alineamento ae es una función* $ae : A \times A_R \mapsto precision \in \Re \in [0,1] \times recall \in \Re \in [0,1]$ *que asocia un alineamiento* A *y un alineamiento de referencia* A_R *a dos números reales que indican la precisión y la cobertura de* A *en relación a* A_R*.*

Definición 6 (Función de Meta-Matching). *Una función de Meta-Matching* mmf *es una función* $mmf : SC \mapsto om$ *que define como se tiene que calcular un determinado valor de simulitud* $sc_i \in SC$*. El resultado debe ser una función de matching optimizada..*

4 Matching estructural

Esta sección describe el diseño y desarrollo de un algoritmo taxonómico para extraer información acerca de las entidades de las ontologías. El algoritmo puede ayudarnos a decidir si dos conceptos son el mismo pero desde el punto de vista de su localización en la ontología. Este tipo de información puede ser útil en

escenarios de alineamiento de ontologías.

Definición 7 (Taxón). *Un taxón es un nombre que designa a una clase o grupo de clases. Un taxón tiene siempre asignado un rango (o nivel de profundidad) que le sitúa en un nivel particular de un sistema jerárquico y que le permite reflejar relaciones.*

Antes de diseñar el algoritmo, es necesario definir los parámetros que calculará el algoritmo. Estos parámetros se almacenarán es una lista enlazada simple con seis registros en cada nodo: *rango*, *hijos*, *hermanos*, *hermanosizq*, *nombre*. La Tabla 1 muestra el tipo de datos y una breve descripción para cada uno de los registros.

Atributo	Tipo	Descripción
rango	*entero*	*Nivel del taxón actual (comienza con 0).*
hijos	*entero*	*Número de hijos del taxón actual*
hermanos	*entero*	*Número de taxones hermanos*
hermanos_izq	*entero*	*Número de taxones hermanos a la izquierda*
mismo_rango	*entero*	*Número de taxones con la misma profundidad*
nombre	*cadena de texto*	*Nombre del taxón*

Table 1. Nodo de la lista enlazada que almacena la información

4.1 El algoritmo estructural

El algoritmo que proponemos se divide en tres pasos:

1. Convertir la ontología en una taxonomía (Véase Fig. 2)
2. Almacenar la taxonomía obtenida en una estructura de datos (Véase Fig. 3).
3. Realizar los cálculos pertinentes para completar los valores de la estructura anterior (Véase Fig. 4).

Por último, es necesario invocar al algoritmo con los parámetros adecuados (Véase Fig. 5).

La filosofía de nuestro algoritmo es la detección de cambios en los rangos de cada taxón de la taxonomía. De esta forma, es posible llevar la contabilidad de los diferentes tipos de vecinos que tiene un determinado concepto.

Convertir la ontología en una taxonomía. Este primer paso consiste en convertir la ontología en una taxonomía que nos permitirá computar de manera más sencilla los datos relativos a la vecindad de cada concepto perteneciente a la ontología. El procedimiento es el siguiente:

```
procedimiento ont2tax (OntoClase cls, List visitados, int rango)
 inicio
   almacenarEnTax(cls, rango) ; Paso 2
   si (NO visitados.contiene( cls ))
     mientras iterador = cls.SubClases hacer
       OntClase sub := (OntClase) iterador.siguiente
       visitados.añadir(cls)
       ont2tax (sub, visitados, rango + 1)
       visitados.eliminar(cls)
     fin mientras
   fin si
 fin
```

Fig. 2. Procedimiento para convertir una ontología en una taxonomía

```
procedimiento almacenarEnTax(OntoClase cls, int rango)
 inicio
   Elemento e:= nuevo Elemento (rango, 0, 0, 0, 0, cls.obtenerNombre)
   DS.añadir (e)
 fin
```

Fig. 3. Procedimiento para almacenar la ontología en forma de taxonomía

Almacenar la taxonomía en la lista enlazada. El método almacenarEnTax tiene la siguiente interfaz: *almacenarEnTax (rango:* ***entero****, hijos:* ***entero****, hermanos:* ***entero****, hermanosizq:* ***entero****, nombre:* ***cadena****)* donde cada parámetro es el valor para una nueva entrada en la lista enlazada. No obstante, en este segundo paso, sólo conocemos la profundidad (rango) y el nombre de cada taxón, por lo que sólo podemos completar parcialmente los nodos de la lista, es decir, sólo podemos invocar el procedimiento con los parámetros rango y nombre.

Completar el cálculo de los parámetros Con los datos almacenados en forma de taxonomía, ya podemos detectar los cambios en los rangos de los taxones. Teniendo en cuenta dichos cambios podemos obtener información tan valiosa como el número de hijos, hermanos, etc. El algoritmo puede desarrollarse teniendo en cuenta estas siete reglas:

1. Todos los taxones con el mismo rango son taxones del mismo nivel.
2. Una cadena de hermanos es una secuencia consecutiva de taxones del mismo nivel.
3. Un cambio hacia un taxón de un rango menor rompe una cadena de hermanos.
4. Todos los hermanos con una posición menor son hermanos a la izquierda.
5. Dado un taxón, si el siguiente taxón tiene un rango mayor, entonces es un hijo.
6. Una cadena de hermanos sólo puede romperse por un cambio hacia un taxón de menor rango.
7. Un taxón dos o más grados superior (en cuyo caso sería nieto, bisnieto...) a otro dado, no rompe la cadena de hijos del segundo, pero no computa como hijo.

Llamada al algoritmo. Ahora es necesario invocar al algoritmo. Hay que definir el modelo ontológico que se va a procesar (por ejemplo, OWL) y señalar cuál es el nodo a partir del cual se empezará a explorar la ontologías. También se necesita inicializar una lista en la que almacenar los nodos que se han ido visitando. La Figura 6 muestra la porción de pseudocódigo relativa a este paso.

4.2 Ejemplo práctico

Hemos elegido una pequeña ontología sobre cámaras para ilustrar cómo funciona el algoritmo. En la Figura 6 podemos ver una representación taxonómica de la ontología, hemos obtenido dicha taxonomía tras aplicar el primer paso del algoritmo. Finalmente, en la Tabla 2, podemos observar los valores que se han obtenido. Estos valores se han conseguido tras aplicar los pasos 2 y 3 del algoritmo.

```
procedimiento completar ()
 variable hijos, hermanos, hermanos_izq:  entero
 variable mismo_nivel, i, j, k, t: entero
 variable bandera: booleano
 inicio
   para i := 0 hasta DS.tamaño
     hijos, hermanos, hermanos_izq := 0
     bandera := falso
     para j := 0 hasta DS.tamaño
       si (j < i)
         si (DS[i].rango = DS[j].rango)
           hermanos++
           hermanos_izq++
         fin si
         si (DS[i].rango ¡ DS[j].rango)
           hermanos := 0
           hermanos_izq := 0
         fin si
       fin si
       si (j > i)
         si (DS[i].rango = DS[j].rango)
           hermanos++
         para si
         ; fin de cadena de hijos
         si (DS[i].rango < DS[j].rango)
           parar
         fin si
       fin si
       ; cómputo de hijos
       si ((j = i+1) Y (DS[i].rango = DS[j].rango - 1) Y (NO bandera))
         para k := j hasta DS[j].rango < DS[k].rango
           si (DS[j].rango = DS[k].rango)
             hijos++
             bandera := cierto
           fin si
         fin para
       fin si
     fin para
     para t := 0 hasta DS.tamaño
       si (NO t=i) Y (DS[i].rango = DS[t].rango)
         mismo_nivel++
       fin si
     fin para
     DS[i].añadirNumHijos (hijos)
     DS[i].añadirNumHermanos (hermanos)
     DS[i].añadirNumHermanosALaIzquierda (hermanos_izq)
     DS[i].añadirNumMismoNivel (mismo_nivel)
   fin si
fin para
```

Fig. 4. Procedimiento para reordenar y completar la estructura

```
OntoModelo m := crearModelo
Iterador i := m.Raiz()
mientras i.tieneSiguiente() hacer
  onto2tax((OntClase) i.siguiente(), new Lista(), 0 )
fin mientras
completar ()
```

Fig. 5. Llamada al algoritmo

```
camera:SLR
camera:Money
camera:BodyWithNonAdjustableShutterSpeed
camera:Range
camera:Window
  camera:PurchaseableItem
    camera:Camera
    camera:Digital
  camera:Large-Format
  camera:Lens
  camera:Body
camera:Viewer
  camera:HQ-Viewer
```

Fig. 6. Representación taxonómica de la ontología de las cámaras

Concepto	*Rango*	*Hijos*	*Hermanos*	*Hermanos Izq.*	*Mismo nivel*
SLR	0	0	5	0	5
Money	0	0	5	1	5
BodyWithNonAd.	0	0	5	2	5
Range	0	0	5	3	5
Window	0	4	5	4	5
PurchaseableItem	1	2	3	0	4
Camera	2	0	1	0	1
Digital	2	0	1	1	1
$Large - Format$	1	0	3	1	4
Lens	1	0	3	2	4
Body	1	0	3	3	4
Viewer	0	1	5	5	5
$HQ - Viewer$	1	0	0	0	4

Table 2. Estructura de datos obtenida de la taxonomía de la Figura 6

4.3 Resultados

El propósito de esta sección es mostrar la relativa facilidad con la que se pueden desarrollar matchers taxonómicos que resuelvan situaciones muy comunes en el alineamiento de ontologías. En las siguientes subsecciones vamos a desarrollar tres casos de usos: cómo usar el algoritmo para contar las hojas de la taxonomía, cómo usarlo para obtener una medida de la similitud entre dos ontologías y cómo usarlo como herramienta de apoyo para el alineamiento de entidades.

4.4 Cómputo de las hojas de la taxonomía

Una de los casos de uso más comunes a la hora de alinear conceptos de una ontología, sobre todo en las técnicas basadas en grafos, consiste en obtener información acerca de si dicha clase es una hoja, es decir, se encuentra en un nodo terminal de la ontología. Con nuestra propuesta, resulta sencillo resolver este caso haciendo uso de un pequeño fragmento de código que comprueba los taxones con mayor rango. La Figura 7 muestra está porción de código.

```
variable max, hojas: entero
max := hojas := 0
para i := 0 hasta DS.tamaño
  si (DS[i].rango > max)
    max := DS[i].rango
  fin si
fin para
para j := 0 hasta DS.tamaño
  si (DS[j].rango = max)
    hojas++
  fin si
fin para
devolver hojas
```

Fig. 7. Cómputo de las hojas de la taxonomía

Hemos computado el número de hojas de la taxonomía, pero en realidad, podríamos haber sido capaces de decidir si cada taxón era una hoja o no. En realidad, podríamos cubrir casos de uso para obtener cualquier tipo de información que no se halle en la estructura.

Uso de SIFO para comparar similitudes estructurales SIFO puede usarse para comparar similitudes estructurales entre ontologías.
Definición 8. *Definimos el índice estructural de una ontología como un número entero no negativo que nos indica la suma global de todas las características estructurales de una ontología.*

Es posible usar el algoritmo para extraer índices estructurales de ontologías que permitan medir su similitud estructural. Como mostramos con anterioridad, algunas técnicas de alineamiento usan métodos estadísticos para obtener similitudes estructurales. El resultado de comparar los índices puede ser útil, entre otras cosas, para ajustar la calidad de los mappings generados. La Figura 8 muestra el código asociado:

```
variable acum : entero
acum := 0
para i := 0 hasta DS.tamaño
   acum := acum + DS[i].rango
   acum := acum + DS[i].hijos
   acum := acum + DS[i].hermanos
   acum := acum + DS[i].hermanosizq
   acum := acum + DS[i].mismonivel
fin para
devolver acum
```

Fig. 8. Porción de código para extraer un índice estructural de la ontología

Hemos usado la estructura de datos rellenada por el algoritmo para computar los índices estructurales de ontologías pertenecientes a varios dominios: bibliografía, departamentos, genealogía y gente. Luego los hemos comparado. La Tabla 3 muestra los resultados que hemos obtenido. Obviamente, cuanto mayor es el porcentaje anotado, mayor es la similitud estructural de las ontologías comparadas.

Ontologías	*Similitud estructural*	*Porcentaje*
Bibliografía [42] *vs* [43]	515/6890	7.4%
Departmentos [40]*vs*[41]	4515/57380	7.8%
Genealogía [44] *vs* [45]	180/275	65.4%
Gente [23] *vs* [24]	525/3150	16.6%

Table 3. Comparación de índices de similitud estructural

4.5 Apoyo al alineamiento de conceptos.

Es muy común usar varios tipos de matchers de ontologías para obtener alineamientos más precisos. Por ejemplo, la similitud entre dos conceptos [22] podría venir dada por la siguiente fórmula arbitraria:

$$similitud(c1, c2) = 0.4 \times Levenshtein + 0.4 \times Google + 0.2 \times Nuestra\ propuesta$$

Esta fórmula permite alinear ontologías teniendo en cuenta tres puntos de vista distintos. El algoritmo de Levhenstein [22] calcula la similitud léxica de los conceptos, el algoritmo de Google [38] calcula la similitud lingüística. Usaremos la información extraída por el algoritmo para alinear dos ontologías reales: [32] y [33]. Usaremos la siguiente regla arbitraria:

$$Si(DS.taxon.atributo = DS2.taxon.atributo) \rightarrow similitud = simimilitud + 0.2$$

De esta forma, si dos taxones coinciden en sus 5 atributos, consideraremos que son totalmente equivalentes, y si no coinciden en ninguno, consideraremos que no lo son. La Tabla 4 muestra los resultados globales para el experimento.

Russia1	**Russia2**	**Levhenstein**	**Google**	**Estructural**	**Total**
food	*food*	1.00	1.00	0.40	0.88
traveller	*normal_traveller*	0.56	0.00	0.20	0.26
health_risk	*disease_type*	0.17	0.00	0.40	0.15
time_unit	*time*	0.44	1.00	0.20	0.62
document	*written_document*	0.50	1.00	0.20	0.64
approval	*certificate*	0.10	0.84	0.40	0.46
payment	*means_of_payment*	0.44	0.00	0.00	0.18
monetary_unit	*currency*	0.15	1.00	0.00	0.46
unit	*unit*	1.00	1.00	0.20	0.84
adventure	*sport*	0.11	1.00	0.20	0.49
building	*public_building*	0.53	0.80	0.40	0.61
flight	*air_travel*	0.10	0.80	0.20	0.76
river_transfer	*cruise*	0.21	0.00	0.20	0.12
railway	*train_travel*	0.33	0.00	0.40	0.21
political_area	*political_region*	0.69	0.10	0.20	0.36

Table 4. Resultados finales para el alineamiento

Si consideramos como válidas aquellas correspondencias cuya propabilidad de ser ciertas es mayor al 60%, tendríamos los siguientes mappings: *(food, food)*, (*time_unit, time*), *(document, written_document)*, *(unit, unit)*, *(building, public_building)* y *(flight, air_travel)*.

Para concluir, hemos ejecutado nuestro algoritmo usando un benchmark estándar como entrada. El objetivo es medir la eficiencia de la implementación. El benchmark vuelve a ser el mencionado con anterioridad [25]. Para probar la implementacíón de nuestro algoritmo, hemos usado un Intel Centrino, 1.66 Ghz y 512 MB de memoria volátil. El tiempo resultante no era siempre el mismo (debido al efecto rodaja de tiempo en los sistemas monoprocesador), por lo que hemos obtenido cada tiempo diez veces y hemos calculado su media aritmética.

Ontología	*Número de nodos*	*Tiempo medio (segundos)*
101	33	0.062
102	76	0.312
103	33	0.063
104	33	0.062
201	33	0.047
202	33	0.062
203	33	0.046
204	33	0.063
224	33	0.092
225	33	0.032
226	33	0.031
Bib/MIT	15	0.093
$BibTeX/UMBC$	13	0.031
$Karlsruhe$	56	0.125
$INRIA$	43	0.112

Table 5. Resultados estadísticos para el benchmark de la OAEI

4.6 Discusión

Hemos extendido la Tabla 7 de [26] para comparar la complejidad computacional de algunas de las mejores herramientas de alineamiento. SIFO no es una herramienta, pero puede implementarse de esa forma. Las herramientas incluidas son NOM [27], PROMPT [28], Anchor-PROMPT [29], GLUE [30] and QUOM [31]. Todos los valores de complejidad de la Tabla 7 vienen dados bajo la suposición de que el acceso a la ontología tiene una penalización constante. Como puede observarse, la complejidad de nuestro algoritmo es tan buena como la de las alternativas estudiadas. Por tanto, su inclusión en herramientas de matching no debería suponer una sobrecarga.

Algoritmo	**Complejidad**
NOM	$O(n^2 \cdot log^2 n)$
$PROMPT$	$O(n \cdot log\ n)$
$Anchor - PROMPT$	$O(n^2 \cdot log^2 n)$
$GLUE$	$O(n^2)$
QOM	$O(n \cdot log\ n)$
Nuestra propuesta	$O(n \cdot log\ n)$

Table 6. Comparación de complejidades

As it can be appreciated, the complexity of SIFO is as good as the considered tools. Therefore, it seems to be a good idea to include it in matching tools to suplement other methods.

Como puede apreciarse, la complejidad de SIFo es tan buena como la de las herramientas tenidas en cuenta. Por tanto, parece una buena idea incluirlo como una herramienta para suplementar otros métodos.

Nuestra propuesta, como sucede a menudo en el mundo de la ingeniería, tiene varias ventajas, pero también otras desventajas. Estas son algunas de ella:

Ventajas

- Permite que la información que se obtiene pueda servir de ayuda para la toma de decisiones en escenarios propicios para el matching de ontologías. Como hemos mostrado en el ejemplo, SIFO es capaz de descubrir correspondencias razonables entre ontologías.
- El algoritmo que proponemos es válido para el alineamiento de ontologías, pero también para alinear taxonomías o directorios.
- Su complejidad computacional es la misma que la de las herramientas más eficientes que se han estudiado.

Desventajas

- En relación al alineamiento de ontologías, es necesario combinar SIFO con otras técnicas para obtener resultados satisfactorios.

Mas allá de los resultados numéricos obtenidos, se ha mostrado la relativa facilidad y eficacia con la que se puede usar para resolver todo tipo de casos de uso relativos al alineamiento de ontologías. Lo que redunda en aspectos tales como la escalabilidad y reusabilidad, así como la facilidad para integrarlo con herramientas de matching ya existentes.

Como trabajo futuro, planteamos un estudio acerca del comportamiento del algoritmo para resolver problemas cotidianos en los que intervienen taxonomías, como pueden ser la comparación de sistemas de ficheros de dos ordenadores, el alineamiento de directorios de direcciones web o la interoperabilidad en el ámbito de repositorios de recursos anotados, como es el caso de las bibliotecas digitales, etc.

5 Matching lingüístico

Actualmente existen muchas técnicas y herramientas que tratan de solucionar el problema del alineamiento de ontologías. No obstante, la compleja naturaleza del problema, en la que intervienen factores dependientes del contexto, de los datos y de los propios usuarios que desean integrar conocimiento, hace que las soluciones propuestas no sean completamente satisfactorias. En este sentido, ha aparecido recientemente la distancia de similitud de Google. Una métrica cuyo objetivo es hacer estimaciones acerca de la similitud de dos términos en base a su aparición en los resultados de búsqueda del conocido buscador. Esta sección consiste en un experimento sistemático que extiende dicha métrica para que pueda emplearse en otros motores de búsqueda.

De hecho, estamos especialmente interesados en tres características de la World Wide Web (WWW):

1. Es la base de datos más grande del mundo. Y posiblemente constituye la fuente más valiosa de conocimiento de propósito general.
2. La información (el conocimiento) que almacena está en lenguaje natural y, por tanto, puede ayudar a resolver problemas relacionados con el Procesamiento del Lenguaje Natural (PLN).
3. Ofrece mecanismos simples capaces de separar la información relevante de la información no relevante.

En este sentido, creemos que la contribución más notable de este trabajo es la identificación de las mejores fuentes de conocimiento web para solucionar el problemas del alineamiento de ontologías de manera precisa. De hecho, en [31], los autores afirman: *Presentamos una nueva teoría de similitud entre palabras y frases que se basa en la distancia de información y en la complejidad de Kolmogorov. Para demostrarlo, vamos a usar la World Wide Web (WWW) como base de datos y Google como motor de búsqueda. El método también es aplicable a otros motores de búsqueda.* Ahí es precisamente donde entra en juego nuestra propuesta.

Bajo ninguna circunstancia este trabajo puede considerarse como una prueba de que un motor de búsqueda es mejor que los demás o que la información que proporciona es más precisa. Tan sólo, que las características a la hora de procesar las consultas y la variedad de los contenidos indexados lo hacen más apropiado para dar soporte al alineamiento de ontologías pertenecientes a los dominios estudiados.

Definición 9 (Medida de similitud). *Un medida de similitud sm es una función* $sm : \mu_1 \times \mu_2 \mapsto R$ *que asocia la similitud de dos mappings de entrada* μ_1 *y* μ_2 *a un valor* $sc \in R$ *en el rango [0, 1].*

Un valor de 0 indica una desigualdad total y un 1 una similitud total entre los dos mappings μ_1 y μ_2.

Definición 10 (Hit). *Hit es cada uno de los elementos encontrados por un motor de búsqueda que coincide con las condiciones de búsqueda especificadas. Más formalmente, podemos definir un hit como la función* $hit : \vartheta \mapsto N$ *que asocia un número natural a una palabra (o conjunto de palabras) estimando así su popularidad en términos absolutos en el subconjunto de Internet indexado por el buscador.*

Un valor de 0 indica que la palabra (o conjunto de palabras) no es popular. Y cuanto mayor sea dicho valor, mayor será su popularidad según el buscador.

5.1 Diseño del experimento

Para diseñar nuestro experimento, vamos a usar el siguiente método para medir la similitud semántica de dos términos. Sean $c1$ y $c2$ conceptos que pertenecen a

dos ontologías diferentes, sea $f(c1, c2)$ una función bidimensional de $f : S \times S \mapsto \aleph$, donde S es el conjunto de todas las cadenas de caracteres y sea : un operador de concatenación de términos:

$$f(c1, c2) = \frac{hit(c1 : blankspace : c2) + hit(c2 : blankspace : c1)}{\frac{hit(c1) \times hit(c2)}{\chi}}$$

Podemos definir la función similitud tal como:

$$similitud(c1, c2) = \begin{cases} f(c1, c2) \; si \; 0 \leq x \leq 1 \\ \\ 1 \qquad\qquad si \quad x > 1 \end{cases}$$

Donde χ es un número entero que permite al resultado situarse en un determinado rango. Nosotros usaremos el valor 10000, obtenido de un estudio preeliminar, de modo que el rango de los valores devueltos por la función esté comprendido aproximadamente entre cero y uno.

La medida de similitud nos da una idea del número de veces que dos conceptos aparecen por separado en comparación con el número de veces que aparecen a la vez en el mismo recurso (página, documento, etc...) de Internet.

Hemos elegido de manera arbitraria los siguientes buscadores para realizar el experimento: Google[4], Yahoo[5], Lycos[6], Altavista[7], MSN[8] and Ask [9].

Algunas de las compañías propietarias de estos buscadores no permiten que se ejecuten muchas consultas, esto está considerado como un servicio de minado por el que hay que pagar, por lo que hemos diseñado un pequeño experimento inicial que requiera un número de consultas que no supere los máximos establecidos por persona y día: Para ello vamos a alinear dos ontologías de tamaño pequeño-medio (sobre unos 50 conceptos aproximadamente) que modelan algunos aspectos de la realidad de Rusia [32] y [33]. Hemos escogido estas dos ontologías porque varias herramientas de matching las han usado en el pasado para presentar sus resultados.

Aunque podríamos lanzar una tarea de alineamiento sobre todas las entidades (conceptos, propiedades de objeto, propiedades de tipo de datos e instancias) de las ontologías, vamos a hacerlo sólo sobre los conceptos, con objeto de no rebasar el límite de consultas al que hacemos mención. La comparación se hace siguiendo un esquema de todos con todos, es decir, por cada concepto de la ontología origen se hace una comparación con todos los conceptos de la ontología destino, se considera el par con una probabilidad de ser cierto más grande y si supera un determinado umbral, se ofrece en el alineamiento final. De esta forma, el número total de consultas a efectuar

[4] http://www.google.com
[5] http://www.yahoo.com
[6] http://www.lycos.com
[7] http://www.altavista.com
[8] http://www.msn.com
[9] http://www.ask.com

$$n^{o}\ total\ consultas\ = m \times\ n$$

Donde m y n son el número de conceptos pertenecientes a la ontologías origen y destino respectivamente.

5.2 Evaluación empírica

La Tabla 8 muestra los resultados que hemos obtenidos.

Russia1	Russia2	Google	Yahoo	Lycos	AltaVista	MSN	Ask
food	food	1.00	0.00	0.01	1.00	0.01	0.02
drink	drink	1.00	0.01	0.30	1.00	0.06	0.04
traveller	normal_traveller	0.00	0.00	0.00	0.00	0.00	0.00
health_risk	disease_type	0.00	0.00	0.00	0.00	0.00	0.00
time_unit	time_unit	0.00	0.00	0.00	1.00	0.00	0.00
document	document	1.00	0.00	0.01	1.00	0.01	0.02
approval	certificate	0.84	0.20	0.00	1.00	0.00	0.00
payment	means_of_payment	0.00	0.45	0.00	1.00	0.00	0.00
monetary_unit	currency	0.00	0.42	0.00	1.00	0.00	0.00
unit	unit	1.00	0.00	0.01	1.00	0.03	0.03
adventure	sport	1.00	0.01	0.03	1.00	0.04	0.40
building	public_building	0.40	0.11	0.00	1.00	0.00	0.00
flight	air_travel	0.80	0.15	0.03	1.00	0.02	0.00
river_transfer	cruise	0.00	0.12	0.00	1.00	0.00	0.00
railway	train_travel	0.00	0.98	0.00	1.00	0.00	0.00
political_area	political_region	0.00	0.40	0.00	1.00	0.00	0.00

Table 7. Resultados que hemos obtenido de los diferentes motores de búsqueda

La Tabla 9 muestra una comparación de las correspondencias semánticas obtenidas, donde Min. es el mínimo de todos los valores devueltos por los buscadores estudiados y Max. es el máximo. Incluimos estas dos columnas para que se pueda apreciar claramente el rango entre el que oscilan los resultados devueltos por los buscadores. FOAM [16] y RiMOM[34] son herramientas de matching que han ofrecido muy buenos resultados en el concurso Ontology Alignment Contest [25]. FOAM es un framework de alineamiento basado en heurísticas que ha usado las ontologías de Rusia para publicar sus resultados. Por su parte RiMOM usa una composición de algortimos básicos de matching y fue la herramienta que arrojó mejores resultados en el citado concurso.

Además, es importante remarcar que este experimento fue realizado en Mayo de 2008. Porque la información indexada por los buscadores no es estática.

5.3 Discusión

Los resultados que hemos obtenidos no han sido suficientemente grandes ni pertenecen a dominios lo sucientemente heterogéneos como para ser definitivos,

Russia1	Russia2	Min.	Max.	Arit.	FOAM	RiMOM
food	food	0.00	1.00	0.34	1.00	0.50
drink	drink	0.01	1.00	0.40	1.00	0.71
traveller	normal_traveller	0.00	0.00	0.00	0.00	0.00
health_risk	disease_type	0.00	0.00	0.00	0.00	0.17
time_unit	time_unit	0.00	1.00	0.17	1.00	1.00
document	document	0.00	1.00	0.34	1.00	0.99
approval	certificate	0.00	1.00	0.34	0.00	0.21
payment	means_of_payment	0.00	1.00	0.24	0.00	0.37
monetary_unit	currency	0.00	1.00	0.24	0.00	0.00
unit	unit	0.00	1.00	0.35	1.00	1.00
adventure	sport	0.01	1.00	0.41	0.00	0.01
building	public_building	0.00	1.00	0.25	0.80	0.60
flight	air_travel	0.00	1.00	0.17	0.00	0.62
river_transfer	cruise	0.00	1.00	0.19	0.00	0.21
railway	train_travel	0.00	1.00	0.33	0.00	0.54
political_area	political_region	0.00	1.00	0.23	0.00	0.40

Table 8. Comparación de los mappings obtenidos

sin embargo, pueden darnos una idea acerca del comportamiento de los diferentes motores de búsqueda a la hora de alinear ontologías. De hecho, sí podemos destacar dos características que llaman poderosamente la atención sobre el conjunto de datos obtenidos:

1. Existe una gran disparidad entre los resultados obtenidos por los motores de búsqueda que se han tenido cuenta. Sería especialmente interesante saber por qué.
2. Aunque la medida de similitud fue diseñada para funcionar con Google, parece funcionar mejor con Altavista.

Con respecto a la primera, hemos de fijarnos en cómo tratan los buscadores las palabras idénticas, las palabras con guiones, los antónimos y los sinónimos. Podemos observar unas oscilaciones muy amplias, en muchos casos totalmente opuestas. Esto demuestra , de manera preeliminar, nuestra hipótesis inicial de que hay fuentes de conocimiento web que son más apropiadas que otras, al menos, para el dominio de las ontologías sobre las que se ha realizado el estudio.

Con respecto a la segunda característica, por qué Altavista ofrece unos mejores resultados que Google, creemos que se debe a que a pesar de que Google responde satisfactoriamente a la hora de comparar sinónimos, fracasa a la hora de comparar palabras separadas por un guión (_). Altavista indexa actualmente muchos menos contenidos que Google, pero el tratamiento de las consultas y/o la indexación de contenido relevante para el dominio de Rusia, hacen que pueda ofrecer mejores resultados.

5.4 Trabajo relacionado

Algunos autores han utilizado el Conocimiento Web en su trabajos, o una generalización de él: el conocimiento background [35][36][37]. El conocimiento background engloba todo tipo de fuentes para extraer información: diccionarios, tesauros, motores de búsqueda, etc... Por lo que el Conocimiento Web puede considerarse un subtipo de él.

Por otra parte, estos trabajos presentan metodologías y validaciones para esas metodologías y validaciones para estas metodologías, pero no ofrecen comparaciones estadísticas. Nuestro trabajo puede verse como una extensión de [38][39][40], donde se presentan varias fórmulas y mecanismos para beneficiarse del conocimiento de Google. Nuestro trabajo es similar a estos estudios, pero se centran en aspectos teóricos, nosotros nos hemos centrado en la cara más práctica y ofrecemos un análisis estadístico de un conjunto más grande de fuentes de Conocimiento Web.

6 Matching estadístico

Esta sección trata sobre un experimiento en el que se han comparado el renderizado textual de ontologías para obtener alineamientos más precisos. El experimento que hemos realizado consiste en tres pasos: renderizar de manera textual dos ontologías, comparar el texto obtenido con varios algoritmos para la comparación de texto y usar el resultado obtenido como factor para mejorar los alineamientos entre ellos. Como resultado, hemos obtenido evidencias de que esta técnica nos da una buena medida de similitud de las ontologías y por tanto puede permitirnos mejorar la efectividad del proceso de alineamiento.

6.1 Definición del problema

Definición 11. *El renderizado textual de una ontología es el resultado de imprimir la información contenida en dicha ontología.*

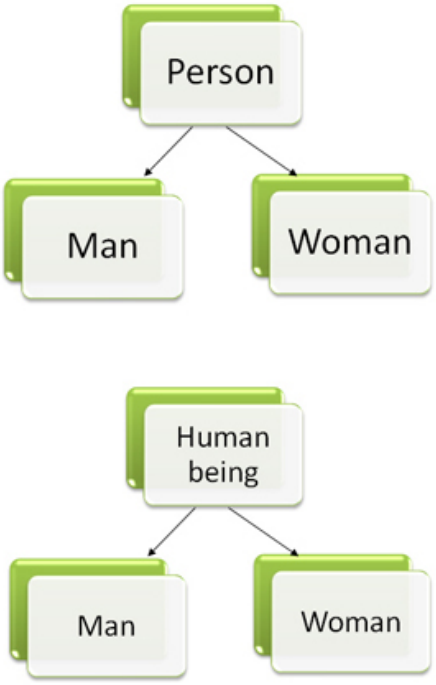
Puede expresarse más formalmente, sea e una entidad de una ontología O y sea $t(e)$ una función que imprime el identificador de una entidad, entonces un renderizado textual T de una ontología O es una expresión tal como:

$$\forall e \in O, \exists t(e) \Rightarrow T(O) = \{t(e)\}$$

Ejemplo 1. El renderizado textual de la Figura 9 es *A man is a person. A woman is a person.*

Ahora vamos a explicar por qué pensamos que los renderizados textuales son interesantes.

Ejemplo 2. Observe las Figuras 9 y 19; son ontologías muy simples, también muy parecidas. Por ejemplo es fácil alinear los conceptos *man* y *woman* usando un algoritmo de comparación de cadenas. Pero qué sucede con *person* y



human being. Sabemos que ambos representan el mismo objeto del mundo real, pero qué algoritmo de matching nos diría que son lo mismo. La técnicas basadas en comparación de cadenas no pueden, los matchers taxonómicos pueden incrementar la probabilidad de que sean lo mismo, pero no es suficeiente; imagine conceptos tales como 'plane' y 'aeroplane', son sinónimos, pero sólo en algunas situaciones. Pensamos que podemos resolver este problema y vamos a hacer un experimento para demostrarlo: Obtengamos el renderizado textual de la primera ontología: *A man is a person. A woman is a person.*

Por otra parte, el renderizado textual de la segunda ontología de ejemplo es: *A man is a human being. A woman is a human being.* Ahora, si comparamos los dos renderizados textuales usando un algoritmo como Loss of Information (LOI) [13], tenemos un 76.9 por ciento de similitud entre ellos. Proponemos usar este resultado como factor para incrementar la probabilidad de que los mappings en el alineamiento de salida sean ciertos.

En este sentido, pensamos que podemos usar esta observación para formular un técnica genérica que mejore los mappings entre ontologías.

El experimento que vamos a realizar consiste en una tarea previa y tres pasos. La tarea previa consiste en lanzar un proceso de alineamiento de ontologías. Resulta interesante lanzar un algoritmo sencillo (del tipo de los basados en comparación de cadenas de texto) para ver cúanto mejoran los siguientes pasos la calidad del alineamiento. Luego hay que hacer:

1. Renderizar las ontologías.
2. Comparar los textos resultante.
3. Usar el resultado como factor para incrementar la probabilidad de que los mappings sean ciertos.

Aunque ya hemos definido el renderizado textual, existen varias formas para renderizar una ontología de manera textual:

Definición 12. *El renderizado crudo es aquel tipo de renderizado que sólo imprime la información de los conceptos y de las propiedades, excluyendo las relaciones. Por lo que pierde información acerca de la estructura.* Este tipo de renderizado es bueno cuando sólo queremos comparar el contenido de las ontologías.

- **Definición 12.1**. *Renderizado crudo parcial es el tipo de renderizado que se usa para medir la similitud entre tipos concretos de entidades entre dos ontologías.* Es útil en casos donde los conceptos son muy parecidos, pero otras entidades (propiedades, relaciones, instancias, etc...) son muy distintas.
- **Definición 12.2**. *Renderizado crudo total es el tipo de renderizado usado para comparar los contenidos de las ontologías completas.* Suele ser útil cuando las ontologías a alinear son muy parecidas.

Definición 13. *Renderizado completo es el tipo de renderizado que permite reconstruir una ontología porque imprime información acerca del contenido y de la estructura. Por lo que es un renderizado sin pérdida de información.* Es útil para comparar no sólo contenidos, sino estructuras.

- **Definición 13.1**. *Renderizado completo pero parcial imprime toda la información relativa a un sólo tipo de entidades.* Como ya comentamos anteriormente, es útil cuando los conceptos son muy parecidos, pero pensamos que las instancias serán muy distintas, por ejemplo.
- **Definición 13.2**. *Renderizado completo imprime toda la información relativa a la ontología, por lo que el proceso es reversible.*

Los renderizados crudos intenta obtener medidas acerca de la similitud de los vocabularios. En los renderizados completos, el parecido de los vocabularios es importante, pero cada vez que una entidad aparece imprimos un mensaje más elaborado. Fíjese en que el mensaje que imprimimos es parecido para las dos ontologías, por lo que estamos incrementando la similitud entre los textos generados, pero también reduciendo la importancia de los vocabularios.

Para obtener resultados empíricos de nuestra teoría, vamos a realizar un experimento sobre dos ontologías pública. Hemos elegido dos ontologías sobre Bibliografía del Institute of Information Sciences (ISI) de California [41] y de la Universidad de Yale [42]. Originalmente, ambas ontologías estaban en formato DAML, pero las hemos convertido a formato OWL para permitir que nuestro software las procese. Las hemos elegido porque suponemos que tienen un alto grado de similitud y por tanto el experimento mostrará las ventajas de nuestra propuesta. Otros detalles importantes que hemos tenido en cuenta son:

– El parámetro R de los mappings (relaciones entre entidades) será sólo de Equivalencia.
– Hemos determinadi que el grado de similitud entre los renderizados textuales será usado para incrementar n (la probabilidad de que la relación sea cierta)

ISI	Yale	n
patent	*Literal*	0.285
collection	*Incollection*	0.833
collection	*Publication*	0.545
booklet	*Incollection*	0.333
booklet	*Book*	0.428
techreport	*Techreport*	0.900
phdthesis	*Inproceedings*	0.307
book	*Book*	0.750
manual	*Literal*	0.285
incollection	*Incollection*	0.916
incollection	*Publication*	0.416
conference	*Incollection*	0.250
proceedings	*Inproceedings*	0.846
inproceedings	*Inproceedings*	0.923
article	*Article*	0.857
inbook	*Incollection*	0.250
inbook	*Book*	0.500

Table 9. Alineamiento de conceptos. Umbral: 0.25

6.2 Resultados

1. En primer lugar, hemos realizado el alineamiento de las ontologías. Hemos usado el algoritmo de Levenshtein [22]. La Tabla 10 muestra los resultados para el alineamiento de conceptos. Hemos determinado un umbral bajo para así obtener un número signficativo de pares. La Tabla 11 muestra los resultados del alineamiento de propiedades. Muchas de ellas están en ambas ontologías.
2. En segundo lugar, hemos realizado el renderizado de las ontologías del ISI y de Yale. Hemos usado un renderizado crudo y completo. De esta forma, estamos dando más importancia a la similitud de los vocabularios que a la estructura de las ontologías.
3. Hemos usado el algoritmo Loss Of Information (LOI) para comparar los textos generados. Hemos obtenido un grado de similitud del 42.2 por ciento.
4. Finalmente, hemos usado ese 42.3 por ciento para incrementar el parámetro n de los mappings (hemos usado la fórmula $n = n + (0.422 \cdot n)$). De esta forma, los valores más altos se incrementan de manera significativa con respecto a los valores de probabilidad más bajos. Las Tablas 12 y 13 muestran los nuevos resultados para los conceptos

ISI	Yale	n
title	*title*	1.000
title	*booktitle*	0.555
note	*note*	1.000
institution	*institution*	1.000
howpublished	*publisher*	0.667
editor	*editor*	1.000
number	*number*	1.000
author	*author*	1.000
volume	*volume*	1.000
location	*Publication*	0.636
year	*year*	1.000
publisher	*publisher*	1.000
mrnumber	*number*	0.750
annote	*note*	0.666
booktitle	*title*	0.555
booktitle	*booktitle*	1.000
edition	*editor*	0.714
organization	*Publication*	0.500
pages	*pages*	1.000
$affiliation$	*Publication*	0.545

Table 10. Alineamiento de propiedades. Umbral: 0.5

En la Tabla 14, hemos extraído un resumen estadístico de los resultados de nuestra propuesta[10]

Como puede ver, al menos en este caso, hemos mejorado la precisión, hemos mantenido la cobertura y obviamente hemos mejorado la medida-F. Pero hay malas noticias también, el número de falsos positivos se ha incrementado. Hemos tenido en cuenta que una relación es verdadera cuando su parámetro n es mayor o igual a 0.9.

Finalmente, hemos repetido el experimento usando ontologías de otros campos: departamentos académicos, personas y genealogía. Como puede ver en la Tabla 15, no podemos determinar todo tipo de relación entre la precisión mejorada y la similitud de los renderizados textuales, pero según los experimentos realizados, la técnica que proponemos es capaz de mejorar la precisión de los mappings.

[10] Hemos usado las siguientes fórmulas pra los cálculos:

$$Precision = \frac{Relacionescorrectas\ relaciones}{Relaciones\ correctas + Relaciones\ incorrectas}$$

$$Recall = \frac{Relaciones\ correctas}{Relaciones\ correctas + Relaciones\ no\ encontradas}$$

$$F - Measure = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

ISI	Yale	n (Improved)
patent	*Literal*	0.405
collection	*Incollection*	**1.000**
collection	*Publication*	0.774
booklet	*Incollection*	0.473
booklet	*Book*	0.608
techreport	*Techreport*	**1.000**
phdthesis	*Inproceedings*	0.436
book	*Book*	**1.000**
manual	*Literal*	0.405
incollection	*Incollection*	**1.000**
incollection	*Publication*	0.591
conference	*Incollection*	0.355
proceedings	*Inproceedings*	**1.000**
inproceedings	*Inproceedings*	**1.000**
article	*Article*	**1.000**
inbook	*Incollection*	0.355
inbook	*Book*	0.711

Table 11. Alineamiento de conceptos mejorado. Umbral: 0.25

ISI	Yale	n (Improved)
title	*title*	1.000
title	*booktitle*	0.788
note	*note*	1.000
institution	*institution*	1.000
howpublished	*publisher*	**0.946**
editor	*editor*	1.000
number	*number*	1.000
author	*author*	1.000
volume	*volume*	1.000
location	*Publication*	0.903
year	*year*	1.000
publisher	*publisher*	1.000
mrnumber	*number*	**1.000**
annote	*note*	**0.946**
booktitle	*title*	0.788
booktitle	*booktitle*	1.000
edition	*editor*	1.000
organization	*Publication*	0.710
pages	*pages*	1.000
affiliation	*Publication*	0.774

Table 12. Alineamiento de propiedades mejorado. Umbral: 0.5

	Before	Later
Precision	63.1%	79.1%
Recall	92.3%	92.3%
$F - Measure$	74.9%	86.5%

Table 13. Resumen del experimento

Ontologies	Similarity	Precision
Departments [40] *vs* [41]	14.8%	+12.5 *p.p.*
People [23] *vs* [24]	19.2%	+8.3 *p.p.*
Bibliography [42] *vs* [43]	42.2%	+16.0 *p.p.*
Genealogy [44] *vs* [45]	61.2%	+7.6 *p.p.*

Table 14. Resultados obtenidos para experimentos en otros dominios

6.3 Discusión

Fíjese en que hay muchos conceptos y propiedades que podrían alinearse usando un algoritmo de normalización de cadenas. No obstante, hay algunos pares que no podrían. Por ejemplo: *proceedings* y *Inproceedings*, *mrnumber* y *number*, *collection* y *Incollection*, etc. Por tanto las ventajas son que hemos tenido en cuenta la similitud de las ontologías para mejorar los mappings. De esta forma, podemos enriquecer los resultados generados por métodos simples. Ofrecemos varias formas de hacerlo: dando más importancia al vocabulario o dando más importancia a la ontología entera. Además, es posible tener en cuenta sólo partes determinadas de la ontología. El resultado de nuestro experimento no muestra que es posible mejorar la precisión y la medida-f del proceso de alineamiento. También hay algunas desventajas, es necesario combinar esta técnica con otras, es decir, no es lo suficientemente buena como para generar mappings por si misma. Además, incrementa el número de falsos positivos. Por otr parte, cabe preguntarse porque no se mejora la cobertura. Piense que nosotros mejoramos unos resultados ya existente. Incrementamos la probabilidad de que los mappings sean ciertos, cuanto más alta sea esta probabilidad, más la incrementamos y viceversa. Pero no hacemos tareas de matching de nuevo. En los experimentos, hemos obtenido un alto grado de similitud, pensamos que este resultado significa que las ontologías comparadas son equivalentes, pero sabemos que hemos alineado ontologías muy parecidas. Tenemos que estudiar en más detalle la forma de enunciar una metodología más precisa.

En esta sección, hemos propuesta una técnica para obtener alineamientos de ontologías más precisos. Esta técnica está basada en la comparación de los renderizados textuales de las ontologías que se desean alinear. De acuerdo con los experimentos que hemos realizado, podemos concluir que la comparación de los renderizados textuales de las ontologías a alinear es capaz de mejorar la precisión del proceso. No obstante, aún queda trabajo por hacer: En primer lugar, es necesario probar un mayor número de ontologías, en particular, sería deseable comprobar el método en las ontologías propuestas por la Ontology Alignment Evaluation Initiative (OAEI) [25]. Además, es importante determinar de manera clara el tipo de renderizado más apropiado de acuerdo con la situación y cuáles son los mejores algoritmos para comparar los renderizados textuales. De esta forma, deseamos no usar sólo el algoritmo LOI, sino otras métricas.

7 Técnicas de Meta-matching

¿Qué es exactamente el Ontology Meta-Matching en la práctica? *Es la técnica para seleccionar los algoritmos, pesos y umbrales apropiados a cada escenario para el alineamiento de ontologías.* La Figura 11 muestra un diagrama para modelar las acciones en un proceso de Meta-Matching.

Fíjese en que los algoritmos no necesitan tenerse en cuenta. La idea es ofrecer la mayor colección de algoritmos posibles y luego el mecanismo se encargará de asociar un peso de 0 a aqueelos que no sean útiles para la resolución del problema. Cómo se usan los algoritmos o cómo se recalculan los pesos y los umbrales de la función de matching es lo que hacen que una determinada estrategia de Meta-Matching sea mejor que otra, en términos de eficacia y eficiencia.

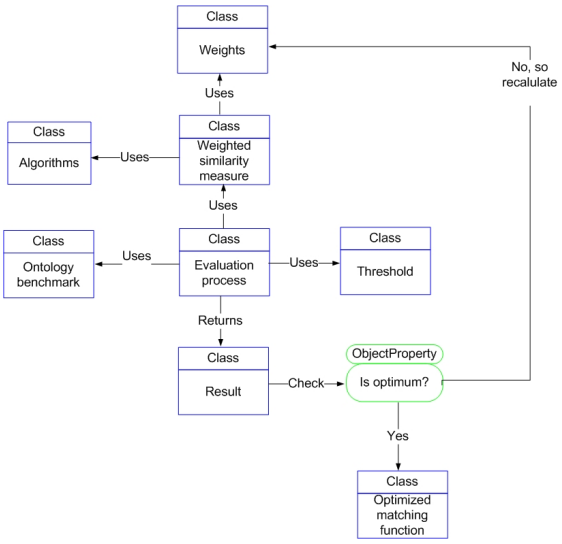


Fig. 9. Modelo general para el Meta-Matching

En general, podemos describir las siguientes características como propias de una tarea de Meta-Matching:

- No es necesario realizarla en tiempo de ejecución.
- Debe ser un proceso automático.
- Debe devolver una única función de matching: la óptima.

Además, el Meta-Matching puede verse desde dos puntos de vista: (i) Desde el punto de vista de las técnicas algorítmicas que se usan para obtener la función de matching:

- **Agregación.** Esta técnica [47] determina los valores T(n) más altos que se obtiene de una secuencia de n algoritmos de matching y luego calcula el valor medio T(n)/n.

- **Combinación.** La idea principal es combinar los valores de similitud devueltos por varios algoritmos de matching para determinar las correspondencias entre entidades de las ontologías. Donde las combinaciones pueden ser tan simples como: media aritméticas o geométricas, máximos, mínimos, distancias de Minkowski o cualquier tipo de producto, suma, etc.. ponderado hasta combinaciones muy complejas como la presentada en Linguistic Combinations [48].
- **Composición** Sean $f_1, f_2, ..., f_n$ una secuencia de n matchers, una composición es una agregación de funciones $f(O_1, O_2) = f_1 \circ f_2 \circ ... \circ f_n$. La idea principal es usar funciones simples para desarrollar funciones más complicadas.

(ii) Desde el punto de vista del paradigma computacional que hace posible el Meta-Matching, por ejemplo, la forma de recalcular los parámetros. Aunque este problemas puede resolverse mediante una técnica de fuerza bruta cuando el número de matchers en bajo, el Meta-Matching escala mejor para un número mayor de matchers. Por esta razón, no incluimos los métodos de fuerza bruta como una técnica viable. Los dos grandes grupos de técnicas que consideramos son:

- **Meta-Matching Heurístico**, donde la aproximación más notable es la basada en Algoritmos Genéticos. En tal caso, se dice que los parámetros se optimizan y la basada en Algoritmos Voraces, en tal caso, se dice que los parámetros se estiman.
- **Meta-Matching basado en Machine-Learning**, donde las aproximaciones más sobresalientes son el Relevance Feedback y las Redes Neuronales. En ambos casos, se dice que los parámetros son aprendidos por el sistema.

7.1 Meta-Matching Heurístico

Una heurística es un método que ayuda a solucionar un problema de manera informal. Se suele usar cuando se pretende conseguir una solución en un tiempo y con una precisión razonables.

Los objetivos fundamentales de la informática son encontrar algoritmos con buenos tiempos de ejecución y con soluciones óptimas. Un algoritmo heurístico ha de renunciar a uno de los dos objetivos; por ejemplo, puede encontrar buenas soluciones, pero sin prueba alguna de que sean las óptimas pero con un tiempo de ejecución razonable.

Por tanto, el uso de heurísticas es muy común en las implementaciones del mundo real. Para muchos problemas prácticos, un algoritmo heurístico es la única manera de obtener buenas soluciones en una cantidad de tiempo razonable.

Hay muchas herramientas que implementan Meta-Matching Heurístico. Se puede ver el ejemplo más claro en la configuración de COMA [13], donde un experto ha ajustado inicialmente los pesos de las técnicas conceptuales y estructurales respectivamente. Para evitar la intervención humana en esta campo, podemos usar los Algoritmos Genéticos para optimizar los parámetros o los Algoritmos Voraces para estimarlos.

Métodos basados en Algoritmos Genéticos. Los Algortimos Genéticos [49] son algoritmos de búsqueda heurística adaptativos basados en las ideas de la selección natural y la genética. El concepto básico de un Algoritmo Génetico es que está diseñado para simular un sistema de evolución natural.

Esta aproximación es capaz de trabajar con varios objetivos [50]: maximizar la precisión, la cobertura o la medida-f así como minimizar la tasa de errores. Además, se ha probado usando algunas medidas de similitud destacadas sobre un benchmark estándar y los resultados que hemos obtenido muestran varias ventajas.

Otra propuesta es [51], una aproximación basada en algortimos genéticos para solucionar el problema del alineamiento de ontologías. Trabaja con una aproximación global-local y conjuntos de características asociadas a cada ontología. Después usa una estrategia heurística de búsqueda, donde la función de fitness es una medida de similitud global entre dos ontologías que está basada en conjuntos de características.

Meta-Matching Voraz. El Meta-Matching voraz [52] es un técnica que dada una tarea de matching determinada, intenta configurar de manera automática una función de ontology matching. Para este propósito, intenta seleccionar los mejores matchers y parámetros que se usarán, por lo que es una estrategia corta de vista. El ejemplo más conocido de este tipo de técnicas puede encontrarse en [53]. Los resultados de las técnicas voraces son, en general, que los basados en algoritmos genéticos, pero su tiempo de computación suele ser mucho más bajo también.

7.2 Métodos basados en Machine Learning

Los métodos de Meta-Matching basados en Machine Learning [54] consideran tanto esquemas como instancias de una ontologías. Este tipo de Meta-Matching puede dividirse en dos subtipos[11]:

Relevance Feedback. Este tipo de aproximaciones explora las validaciones del usuario sobre los alineamientos iniciales para optimizar los parámetros de configuración y las estrategias de matching. El más claro ejemplo de este tipo de Meta-Matching es [55]. El uso de este tipo de técnicas hace que los alineamientos sean cada vez más satisfactorios para el usuario, no obstante, implica un gasto de tiempo enorme en entrenar al sistema y no es capaz de vencer el problema de la dependencia del usuario. Para evitar estos problemas, pueden usarse Redes Neuronales.

[11] Aunque existen más técnicas de aprendizaje como Aprendizaje Bayesiano, WHIRL, árboles de decisión, etc... No existen propuestas para Meta-Matching en este sentido aún

Entrenamiento de Redes Neuronales. Una red neuronal [56] es un grupo de redes neuronales artificiales interconectadas que usan modelos computacionales o matemáticos para procesar la información basándose en el concepto de conexión. En la mayoría de los casos, una red neuronal es un sistema adaptativo que cambia su estructura en base a información externa o interna que fluye a través de la red. En términos más prácticos, las redes de neuronas son modelos de datos estadísticos y no lineales que se usan en la toma de decisiones. Pueden usarse para modelar relaciones muy complejas en base a las entradas y salidas y de esta forma pueden encontrar patrones de datos.

El entrenamiento de redes neuronales para propósitos de Meta-Matching consiste en entrenar redes neuronales con benchmarks lo suficientemene heterógeneos para que luego la red neuronal pueda enfrentarse a problemas del mundo real. Este es el caso de [57] donde los autores explotan una aproximación para aprender los pesos de los diferentes aspectos semánticos de un ontología mediante el uso de una técnica de redes de neuronas artificiales.

Otra aproximación consiste en un método de alineamiento automático basado en un modelo de redes neuronales recursivas que usa instancias de las ontologías para aprender las similitudes que existen entre conceptos. Las redes neuronales recursivas son una extensión del modelo común de redes neuronales, pero que se han diseñado para procesar de manera eficiente datos estructurados [58].

8 Estrategia voraz

En esta sección, vamos a discutir un estrategia voraz para resolver el problema del Ontology Meta-Matching. Además, proponemos un algoritmo voraz eficiente y ofrecemos el cálculo de la complejidad que lleva asociada según la notación O.

8.1 Maximum Similarity Measure (MaSiMe)

Una aproximación ideal para una medida de similitud parametrizable podría definirse de la siguiente forma:

Sea $\boldsymbol{A}$ un vector de algoritmos de matching simples que vienen definidos en forma de medidas de similitud y sea $\boldsymbol{w}$ un vector de pesos numéricos:

$$MaSiMe(c1, c2) = x \in [0, 1] \in \Re \rightarrow \exists \left\langle \boldsymbol{A}, \boldsymbol{w} \right\rangle, x = max(\sum_{i=1}^{i=n} A_i \cdot w_i)$$
$$\textit{with the following restriction } \sum_{i=1}^{i=n} w_i \leq 1$$

Pero desde el punto de vista de la ingeniería, esta medida conduce a un problema de optimización a la hora de calcular el vector de pesos, porque el número de candidatos del espacio de soluciones es infinito. Por esta razón, presentamos MaSiMe, que usa la noción de granularidad para determinar un número finito de soluciones pertenecientes al espacio de soluciones. Esta solución hace que el problema del cálculo del vector de pesos pueda resolverse en tiempo polinomial.

Definición 14. Maximum Similarity Measure (MaSiMe).

$$MaSiMe(c1, c2) = x \in [0, 1] \in \Re \rightarrow \exists \langle \boldsymbol{A}, \boldsymbol{w}, g \rangle, x = max(\sum_{i=1}^{i=n} A_i \cdot w_i)$$
$$\textit{con las siguientes restricciones } \sum_{i=1}^{i=n} w_i \leq 1 \wedge \forall w_i \in \boldsymbol{w}, w_i \in \{\dot{g}\}$$

Ejemplo 3. Dado un vector arbitrario de algoritmos y una granularidad de 0.05, calcular MaSiMe para el par *(author, name_author)*.

$$MaSiMe(author, name\_author) = .542 \in [0, 1] \rightarrow$$
$$\exists \langle A = (L, B, M, Q), w = (0.8, 0, 0, 0.2), g = 0.05 \rangle, 0.542 = max(\sum_{i=1}^{i=4} A_i \cdot w_i)$$

Donde L = Levhenstein [22], B = BlockDistance [13], M = MatchingCoefficient [13] , Q = QGramsDistance [58]

Hay varias propiedades para esta definición:

Propiedad 1 (Distribución uniforme y continua). A priori, MaSiMe presenta una distribución uniforme y continua en el intervalo $[0, 1]$, es decir, su función de densidad probabilística está caracterizada por:

$$\forall\ a, b \in [0, 1] \rightarrow f(x) = \tfrac{1}{b-a}\ for\ a \leq x \leq b$$

Propiedad 2 (Maximalidad). Si uno de los algoritmos pertenecientes al vector de algoritmos de matching devuelve una similitud de 1, el valor de MaSiMe es 1.

$$\exists A_i \in \boldsymbol{A},\ A_i(c1, c2) = 1 \rightarrow MaSiMe(c1, c2) = 1$$

Además, el recíproco es cierto

$$MaSiMe(c1, c2) = 1 \rightarrow \exists A_i \in \boldsymbol{A},\ A_i(c1, c2) = 1$$

Ejemplo 4. Supongamos que tenemos el vector: : $\boldsymbol{A}$ = *(Google Similarity Distance [38], BlockDistance, MatchingCoefficient, QGramsDistance)* y $g = 0.05$, calcule $\boldsymbol{w}$ para maximizar R en el mapping *(plane, aeroplane, Equivalencia, R) Soluciónm*:

$$(1, 0, 0, 0)$$

Por lo que el matcher óptimo para el mapping *(plane, aeroplane)* es:

$$1 \cdot GoogleDistance + 0 \cdot BlockDistance + 0 \cdot MatchingCoefficient + 0 \cdot$$
$$QGramsDistance, R = 0.555$$

Además, podemos decir que el peor vector es $\boldsymbol{w} = (0, 0.8, 0.15, 0.05)$ porque genera $R = 0.027$

Propiedad 3 (Monoticidad). Sea S un vector de algoritmos de matching, sea S' un superconjunto de S. Entonces, Si MaSiMe tiene un valor específico para S, luego el valor para S' es mayor o igual que dicho valor.

$$\forall S' \supset S, MaSiMe_s = x \rightarrow MaSiMe_{s'} \geq x$$

Propiedad 4. (Completitud dependiende). Si uno de los algoritmos que pertenece al conjunto de algoritmos de matching ofrece una similitud de 1 y la granularidad elegida no es submúltiplo de 1, el valor de MaSiMe será inferior a 1.

$$\exists A_i \in \boldsymbol{A} \wedge \ 1 \notin \{\dot{g}\} \wedge A_i(c1, c2) = 1 \rightarrow MaSiMe(c1, c2) < 1$$

Ejemplo 5. Supongamos las mismas condiciones del Ejemplo 4, es decir, que tenemos A = *(Google Similarity Distance, BlockDistance, MatchingCoefficient, QGramsDistance)* pero ahora $g = 0.21$. Calcule el $\boldsymbol{w}$ para maximizar R en el mapping *(plane, aeroplane, Equivalencia, R)*
Solución:

$$(0.84, 0, 0, 0)$$

Por lo que el matcher óptimo para *(plane, aeroplane)* no es el mismo que en el Ejemplo 4.

La razón es que la granularidad no es múltiplo de 1, el sumatorio del vector de pesos no puede ser 1 y por tanto $A \cdot \boldsymbol{w}$ no puede ser óptimo.

8.2 Calculando el vector de pesos

Una vez que el problema está claro y los parámetros $\boldsymbol{A}$ y g se conocen, es necesario calcular de manera eficiente el vector de pesos. Llegados a este punto, dejamos el campo de las medidas de similitud para buscar una solución desde el punto de vista de la ingeniería.

Es posible calcular MaSiMe de varias formas, en este trabajo, hemos diseñado un mecanismo voraz que parece ser efectivo y eficiente. Los próximos párrafos describen dicho mecanismo.

El algoritmo Un algoritmo voraz es un algoritmo que resuelve un problema pero que probablemente lo haga en base a un óptimo local. Sea S el conjunto de todos los algoritmos para el matching de ontologías:

$$\exists A \subset S, \exists g \in [0,1] \in Q, \forall i, j, k, ..., t \in \{\dot{g}\} \rightarrow \exists \boldsymbol{r}, r_i = \boldsymbol{A} \cdot (i, j-i, k-j, ..., 1-t)$$
$$\textit{with the followings restrictions } j \geq i \wedge k \geq j \wedge 1 \geq k$$
$$R = max\ (r_i) \leq 1$$

Donde,

- g es la granularidad
- $(i, j-i, k-j, ..., 1-t)$ es el patrón para el vector de pesos
- r_i son los resultados parciales
- R es el máximo de los valores parciales, y por tanto el valor de MaSiMe

El algoritmo puede suspenderse cuando se obtiene un resultado parcial igual a 1, porque es el valor máximo que podemos esperar.

Complejidad. The strategy seems to be brute force, but it is not. Have into account that the input data size is $n^{lenght\ of\ A}$, but the computational complexity for the algorithm according to O notation is

La estrategia se parece a la fuerza bruta, pero no es. Tenga en cuenta que el tamaño de los datos de entrada es

$$O(n^{lenght\ of\ A-1})$$

De esta forma, la complejidad total (TC) para MaSiMe es:

$$TC(MaSiMe_A) = O(max(max(O(A_i)), O(strategy)))$$

y por tanto, MaSiMe usando la estrategia voraz:

$$TC(MaSiMe_A) = O(max(max(O(A_i), O(n^{lenght\ of\ A-1})))$$

Ejemplo 6. Dado el conjunto $A = \{$*Levhenstein, BlockDistance, MatchingCoefficient, QGrams-Distance*$\}$, la complejidad del proceso de matching usando MaSiMe se calcula:

$$TC(MaSiMe_A) = O(max(O(n^2), O(n^3))) = O(n^3)$$

8.3 Evaluación empírica

En esta sección, probamos una implementación de MaSiMe. Hemos configurado MaSiMe de la siguiente forma: Para el vector de algoritmos de matching, hemos elegido un conjunto arbitrario de algoritmos $A = \{$*Levhenstein [22], Stolios [59], SIFO [60], Google [38] [39], Q-Gram [58]* $\}$ y para la granularidad, g = 0.05. Además, el umbral para los mappings relevantes lo hemos definido en 0.9, es decir, todos los mappings con un probabilidad mayor a 0.9 de ser ciertos, serán incluidos en el alineamiento de salida.

Antes de probar MaSiMe sobre un benchmark estándar, mostramos un ejemplo de las relaciones de equivalencia que MaSiMe es capaz de descubrir entre las ontologías [17] y [18]. Es un ejemplo arbitrario, pero nos da una idea de cómo se comporta la medida. La hemos comparado con dos herramientas muy destacadas dentro del mundo del alineamiento FOAM [16] and RiMOM [34].

Hemos usado la configuración por defecto para estas herramientas, pero no para MaSiMe, donde la noción de configuración no existe. La Tabla 16 muestra los resultados que hemos obtenido:

9 Estrategia basada en Algoritmos Genéticos

Los Algoritmos Genéticos (AAGG) se suelen usar para buscar soluciones en espacios con muchas dimensiones. Por ejemplo, si queremos encontrar el máximo valor de la función wsf con tres variables independientes x, y y z:

Russia1	Russia2	FOAM	RiMOM	MaSiMe
food	food	1.00	0.50	1.00
drink	drink	1.00	0.71	1.00
traveller	normal_traveller	0	0	0.90
health_risk	disease_type	0	0.17	0.17
time_unit	time_unit	1.00	1.00	1.00
document	document	1.00	0.99	1.00
approval	certificate	0	0.21	0.96
payment	means_of_payment	0	0.37	0.86
monetary_unit	currency	0	0	0.19
inhabitant	cititzen_of_russia	0	0.11	0.11
unit	unit	1.00	1.00	1.00
adventure	sport	0	0.01	0.10
building	public_building	0.80	0.60	0.90
flight	air_travel	0	≈ 0	0.62
river_transfer	cruise	0	0.21	0.21
railway	train_travel	0	0.54	1.00
political_area	political_region	0	0.40	0.84

Table 15. Comparación de distintas herramientas de matching

$$wsf(O_1, O_2) = \\ x \cdot datatype(O_1, O_2) + y \cdot normalization(O_1, O_2) + z \cdot synonyms(O_1, O_2)$$

donde x, y y z son pesos que determinan la importancia de las tres medidas de similitud asociadas y que pertenecen al intervalo real [0, 1]. El problema que nosotros queremos resolver es encontrar los valores de x, y y z que maximizan el valor de wsf

Mientras este problema puede resolverse de manera trivial mediante métodos de fuerza bruta sobre el rango de las variables independientes x, y y z, los AAGG escalan mejor para problemas de una mayor dimensionalidad; por ejemplo cuando el número de variables independientes pasa a ser x, y, z,..., t. En este caso, una búsqueda que abarque todo el espacio de soluciones sería demasiado costosa.

La metodología de aplicación de estos algoritmos necesita que se definan las siguientes dos características:

- La caracterización del problema mediante la codificación en cadenas de bits de las posibles soluciones.
- La definición de un función de fitness que permita evaluar la relativa calidad de cada solución que pertenezca a la población.

Nuestra primera tarea consiste en caracterizar el espacio de búsqueda con algunos parámetros. Necesitamos codificar varios parámetros en un cromosoma, por lo que hemos diseñado un método para convertir una representación de 10 bits en un conjunto de números en coma flotante en el rango [0, 1].

Luego hemos diseñado una función de fitness para determinar los cromosomas de la población que merecen vivir y reproducirse mediante operaciones de cruce y mutación.

Para el valor devuelto por la función de fitness, podemos seleccionar los parámetros devuelto por el proceso de evaluación de un alineamiento, es decir, precision, recall, la f-measure o fall-out. De esta forma, estamos ofreciendo la posibilidad de guiar al sistema hacia la optimización de dichos parámetros:

- Optimización de la precisión
- Optimización de la cobertura
- Optimización de la medida-f
- Reducción de la tasa de errores

Todos estos conceptos se usan en el campo de la Recuperación de la Información [62] para medir la calidad de una tarea de recuperación. La precisión es el porcentaje de elementos devueltos que son relevantes. La cobertura es la fracción de elementos que son relevantes para una consulta (en este caso una tarea de matching). La medida-f es una suma ponderada de la precisión y la cobertura. Finalmente, la tasa de errores es el porcentaje de mappings que se ofrecen al usuario sin ser ciertos. En algunos dominios, por ejemplo en Medicina, los errores están absolutamente prohibidos.

9.1 Estudio preeliminar

Vamos a hacer un estudio preeliminar de los parámetros del algoritmo.

- Para el número de genes por cromosoma, hemos seleccionados los valores 5, 10 y 20. Un estudio de la distribución T-Test nos ha mostrado que las diferencias entre las muestras no son estadísticamente significativas. Por tanto, hemos seleccionado 20 genes por cromosoma.
- Para el número de individuos en la población, hemos seleccionados los valores de 20, 50 y 100. De nuevo, la distribución estadística T-Test nos ha mostrado que las diferencias entre muestras no son estadísticamente significativas. Por lo que hemos seleccionado una población de 100 individuos.
- En relación con las fracciones cruce y mutación, hemos elegido un valor alto para la tasa de cruces entre genes y un pequeño porcentaje de población para las mutaciones con el objeto de buscar más allá de los óptimos locales. Esta es la configuración clásica de un algoritmo genético de tipo elitista.
- Después de diez ejecuciones independientes, nos hemos dado cuenta de que el algoritmo genético no mejora los resultados más allá de la quinta generación, por lo que hemos fijado el número máximo de generaciones en cinco.

9.2 Experimento principal

Related to the conditions of the experiment, we have used:

En relación a las condiciones del experimento, hemos usado:

- Como vector de medidas de similitud:
  {*Levhenstein*[22], *SIFO*[60], *Stolios*[59], *QGrams*[58]}
- The GA has been configured having into account the following parameters[12]:
  El Algoritmo Genético se ha configurado teniendo en cuenta los siguientes parámetros[13]:
  - 20 genes por cromosoma
  - Una población de 100 individuos
  - 0.98 para la tasa de cruce
  - 0.05 para el porcentaje de mutación
  - Hasta un número máximo de 5 generaciones
- Las características de la plataforma son: Intel Core 2 Duo, 2.33Ghz y 4GB RAM. El lenguaje de programación ha sido Java.

10 Evaluación

La evaluación de una estrategia de Meta-Matching consiste en la evaluación de la función de matching que devuelve.

Hay varias formas de evaluar esta función:

- Medidas que dan una idea acerca de la calidad de los mappings identificados
- Medidas acerca del rendimiento en términos de tiempo y memoria consumidos
- Medidas en función del usuario, que intentan averiguar la satisfación del usuario
- Medidas altamente dependientes del caso de aplicación

En la práctica, no obstante, hay cierto grado de consenso a la hora de usar medidas provenientes del campo de la Recuperación de la Información [61]. Entre ellas destacan: *precision*, *recall*, *f-measure* y *fall-out*.

$$Precision = \frac{\{mappings\ relevantes\} \cap \{mappings\ devueltos\}}{\{mappings\ relevantes\}}$$

$$Recall = \frac{\{mappings\ relevantes\} \cap \{mappings\ devueltos\}}{\{mappings\ devueltos\}}$$

$$F - Measure = \frac{2 \times precision \times recall}{precision + recall}$$

$$Fall - out = \frac{\{mappings\ no\ relevantes\} \cap \{mappings\ devueltos\}}{\{mappings\ no\ relevantes\}}$$

La Tabla 17 muestra los resultados que hemos obtenido para la estrategia voraz. La Tabla 18 muestra los resultados que se obtuvieron al aplicar la estrategia basada en algortimos genéticos. La Figura 12 muestra una gráfica comparativa entre las dos anteriores estrategias.

[12] Fitness and search space have been explained in the previous section

[13] La función de fitness y el espacio de búsqueda se han explicado en la subsección previa

Ontología	Comentario	Precision	Recall	F-Meas.	Fall-out
101	Reference alignment	1.00	1.00	1.00	0.00
102	Irrelevant ontology	N/A	N/A	N/A	N/A
103	Language generalization	1.00	1.00	1.00	0.00
104	Language restriction	1.00	1.00	1.00	0.00
201	No names	1.00	1.00	1.00	0.00
202	No names, no comments	1.00	1.00	1.00	0.00
203	Comments was missspelling	1.00	1.00	1.00	0.00
204	Naming conventions	1.00	0.91	0.95	0.00
205	Synonyms	1.00	0.19	0.33	0.00
206	Translation	1.00	0.19	0.33	0.00
221	No specialisation	1.00	1.00	1.00	0.00
222	Flatenned hierarchy	1.00	1.00	1.00	0.00
223	Expanded hierarchy	1.00	1.00	1.00	0.00
224	No instance	1.00	1.00	1.00	0.00
225	No restrictions	1.00	1.00	1.00	0.00
301	Real: BibTeX/MIT	0.93	0.23	0.37	0.06

Table 16. Comportamiento de MaSiMe para el benchmark estándar de la OAEI

Ontología	Comentario	Precision	Recall	F-Meas.	Fallout
101	Reference alignment	1.00	1.00	1.00	0.00
102	Irrelevant ontology	N/A	N/A	N/A	N/A
103	Language generalization	1.00	1.00	1.00	0.00
104	Language restriction	1.00	1.00	1.00	0.00
201	No names	1.00	1.00	1.00	0.00
202	No names, no comments	1.00	1.00	1.00	0.00
203	Comments was missspelling	1.00	1.00	1.00	0.00
204	Naming conventions	1.00	1.00	1.00	0.00
205	Synonyms	1.00	0.71	0.83	0.06
206	Translation	1.00	1.00	1.00	0.00
221	No specialisation	1.00	1.00	1.00	0.00
222	Flatenned hierarchy	1.00	1.00	1.00	0.00
223	Expanded hierarchy	1.00	1.00	1.00	0.00
224	No instance	1.00	1.00	1.00	0.00
225	No restrictions	1.00	1.00	1.00	0.00
301	Real: BibTeX/MIT	0.90	0.69	0.78	0.07

Table 17. Comportamiento del algoritmo genético para el benchmark de la OAEI

11 Trabajo relacionado

Si acudimos a la literatura, podemos distinguir entre algoritmos para el matching de ontologías simples (FCA-MERGE [63] o S-Match [64]) que aplican un sólo método de matching a los elementos, por ejemplo, métodos estructurales o métodos lingüísticos. O métodos que combinan varios métodos, llamadas solucines compuestas, cuya misión es intentar vencer las limitaciones de los algoritmos simples. Una solución compuesta sigue el paradigma de caja negra, en el que varios algoritmos simples dan lugar a un nuevo algoritmo, la forma de composición es dependiente del usuario. Como ejemplos destacan: COMA++ [13], RIMOM [22], FALCON [65] and CtxMatch [66].

Fig. 10. Comparación de resultados obtenidos según la estrategia

El problema de este tipo de propuestas es que usan pesos determinados por un experto para configurar los algoritmos, pero nuestra propuesta calcula de manera automática dichos pesos, por lo que el proceso puede ser más rápido y preciso.

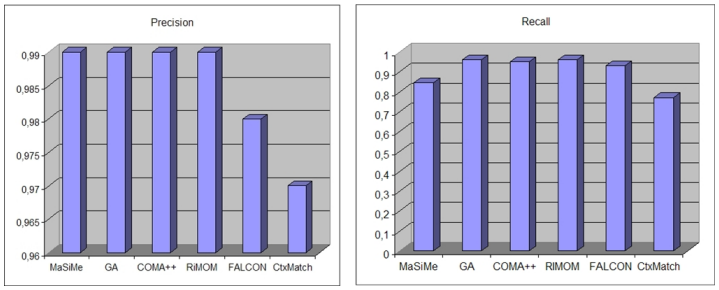


Fig. 11. Comparación con otras herramientas

To avoid the human expert intervention, there are two research lines now; one for evaluating the results of an alignment tool and maybe feedback the

process [67] and another called Ontology Meta-Matching [2] that tries to optimize automatically the parameters related to matching task. So, our approach could be considered a mechanism for Meta-Matching. Most outstanding examples for this paradigm are: (i) Based on Exhaustive search solutions, (ii) Based on Neural Networks solutions, and (iii) Based on Genetic Algorithms solutions:

Para evitar la intervención de un experto humano, existe dos líneas de investigación: una que evalua los resultados de una herramienta de alineamiento y retroalimenta el proceso, cuyo ejemplo más destacado es [67] y otra llamada Ontology Meta-Matching [2] que intenta optimizar de manera automática los parámetros de la tarea de matching. Nuestra propuesta puede considerarse como un mecanismo para Meta-Matching. Los ejemplos más sobresalientes para este paradigma son: (i) Soluciones basadas en Algoritmos Voraces, (ii) Soluciones basadas en Redes Neuronales y (iii) Soluciones basadas en Algoritmos Genéticos:

11.1 Soluciones basadas en Algoritmos Voraces

El problema del Ontology Meta-Matching puede solucionarse buscando por todo el espacio de soluciones cunado el número de medidas de similitud a componer es bajo, la aproximación más notable en este sentido es eTuner [54], que es un sistema que dada una tarea de matching determinada, automáticamente calcula la función de matching que mejor se ajusta teniendo en cuenta alineamientos a nivel de entidades.

Sin embargo, este tipo de búsquedas son muy costosas. Además de totalmente inviables desde el punto de vista computacional cuando el número de medidas de similitud a combinar es muy grande. En este sentido, es aconsejable evitar, en la medida de lo posible, este tipo de métodos.

11.2 Soluciones basadas en Machine Learning

Las soluciones basadas en Machine Learning pueden dividirse en dos subtipos: Basadas en Relevance feedback [68] y Basadas en Redes Neuronales [69]:

- La idea que subyace detrás del Relevance FeedBack es tomar los resultados devueltos inicialmente para una consulta dada y usar la información del usuario acerca de su relevancia para mejorar las sucesivas consultas. APFEL (Alignment Process Feature Estimation and Learning) [68] es una propuesta basada en Relevance Feedback que explora las validaciones del usuario sobre los alineamientos iniciales para configurar de manera automática los parámetros del sistema (pesos, umbrales, etc...)
- Las Redes Neuronales [69] son un modelo de datos estadístico y no lineal empleado en la toma de decisiones por computador. Pueden usar relaciones muy complejas entre las entradas y las salidas para encontrar patrones en los datos. SFS [70] es una herramienta para Ontology Meta-Matching que intenta obtener de manera automática un vector de pesos para hallar los diferentes aspectos semánticos de la tarea de matching, tales como pueden ser la comparación de los indentificadores de conceptos, la comparación de

propiedades y relaciones, etc... Todo ello está basado en técnicas avanzadas de Redes Neuronales.

No obstante, este tipo de soluciones requieren un gran esfuerzo en términos de tiempo y esfuerzo para entrenar a los sistemas. Característica que los otros dos tipos de soluciones no presentan.

11.3 Soluciones basadas en Algoritmos Genéticos

En relación a las soluciones basadas en Algoritmos Genéticos, la herramienta más sobresaliente es GAOM, un algoritmo genético que intenta dar solución al problema de alinear ontologías. Para este propósito define dos características de las ontologías: la intensional y la extensional. Luego un algoritmo genético trata de hallar la solución óptima.

La Tabla 19 muestra una comparativa de los resultados obtenidos por GAOM y por nuestra propuesta.

	Precision	**Recall**
GAOM	0.94	0.87
GOAL	**0.99**	**0.96**

Table 18. Comparación entre GAOM y GOAL

Aunque ambas siguen el mismo paradigma, nuestra propuesta es ligeramente mejor en términos números que GAOM como muestran los resultados.

12 Conclusiones

Hemos presentado el Ontology Meta-Matching como una disciplina prometedora para realizar de manera flexible y precisa alineamientos de ontologías y que generaliza y extiende propuestas previas para hacer uso de algoritmos de matching simples. Hemos presentado las principales técnicas para Ontology Meta-Matching. Estas técnicas tienen en cuenta que no es trivial determinar los pesos asociados a los aspectos semánticos de un alineamiento e intentan evitar el trabajo de investigación encaminado al desarrollo de soluciones basadas en expertos.

Hemos ofrecido un análisis de los algoritmos y técnicas más conocidos a la hora de realizar matching simple y hemos caracterizado su aplicabilidad como cajas negras en un entorno de Meta-Matching. Es necesario tener en cuenta que el éxito del Meta-Matching depende en gran medida del tipo de algoritmos simples que le dan soporte y de la heterogeneidad y completitud de los benchmarks usados para hallar las funciones de optimización.

Hemos mostrado las herramientas de Meta-Matching más prometedoras. Al igual que las técnicas, las herramientas pueden basarse en heurísticas y en basadas en aprendizaje. Estas herramientas representan un esfuerzo serio para

conseguir que la tarea de alinear ontologías sea un proceso más independiente de los usuarios, el contexto y los usuarios.

Las lecciones aprendidas en relación al Ontology Met-Matching nos permitirán trabajar con otros tipos de esquemas conceptuales para modelar conocimiento [71]. En este sentido, estamos convencidos de que el Ontology Meta-Matching es el candidato perfecto para llevar a los usuarios un paso más alla en relación a la interoperabilidad dentro de la Web Semántica.

13 Agradecimientos

Agradecemos a todos los revisores anónimos todos los comentarios y sugerencias que han aportado. También queremos agradecer a Enrique Alba su inestimable colaboración en la parte relativa a Algoritmos Genéticos. El trabajo se ha desarrollado en el marco del proyecto ICARO: TIN2005-09098-C05-01 del antiguo Ministerio de Educación y Ciencia de España y del proyecto Aplicaciones para la Biología de Sistemas, P07-TIC-02978 de la Consejería de Innovacion, Ciencia y Empresa de la Junta de Andalucía.

References

1. Jorge Martinez-Gil: Thinking on the Web: Berners-Lee, Gödel and Turing. Comput. J. 50(3): 371-372 (2007).
2. Jerome Euzenat, Pavel Shvaiko. Ontology Matching. Springer-Verlag, 2007.
3. Philip A. Bernstein, Sergey Melnik: Meta Data Management. ICDE 2004: 875.
4. Bin He, Kevin Chen-Chuan Chang: Making holistic schema matching robust: an ensemble approach. KDD 2005: 429-438.
5. Marc Ehrig, York Sure: Ontology Mapping - An Integrated Approach. ESWS 2004: 76-91.
6. Liliana Cabral, John Domingue, Enrico Motta, Terry R. Payne, Farshad Hakimpour: Approaches to Semantic Web Services: an Overview and Comparisons. ESWS 2004: 225-239.
7. A. Prasad Sistla, Clement T. Yu, R. Venkatasubrahmanian: Similarity Based Retrieval of Videos. ICDE 1997: 181-190.
8. Christoph Kiefer, Abraham Bernstein, Markus Stocker: The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks. ISWC/ASWC 2007: 295-309.
9. Jorge Martinez-Gil, Jose F. Aldana-Montes: Reverse ontology matching. SIGMOD Record 39(4): 5-11 (2010).
10. David Urdiales-Nieto, Jorge Martinez-Gil, Jose F. Aldana-Montes: MaSiMe: A Customized Similarity Measure and Its Application for Tag Cloud Refactoring. OTM Workshops 2009: 937-946.
11. WordNet. `http://wordnet.princeton.edu`. Visit date: 11-march-2008.
12. Patrick Ziegler, Christoph Kiefer, Christoph Sturm, Klaus R. Dittrich, Abraham Bernstein: Detecting Similarities in Ontologies with the SOQA-SimPack Toolkit. EDBT 2006: 59-76.
13. Hong Hai Do, Erhard Rahm: COMA - A System for Flexible Combination of Schema Matching Approaches. VLDB 2002: 610-621.

14. David Aumueller, Hong Hai Do, Sabine Massmann, Erhard Rahm: Schema and ontology matching with COMA++. SIGMOD Conference 2005: 906-908.
15. Christian Drumm, Matthias Schmitt, Hong Hai Do, Erhard Rahm: Quickmig: automatic schema matching for data migration projects. CIKM 2007: 107-116.
16. Marc Ehrig, York Sure: FOAM - Framework for Ontology Alignment and Mapping - Results of the Ontology Alignment Evaluation Initiative. Integrating Ontologies 2005.
17. Yannis Kalfoglou, W. Marco Schorlemmer: IF-Map: An Ontology-Mapping Method Based on Information-Flow Theory. J. Data Semantics 1: 98-127 (2003).
18. Haggai Roitman, Avigdor Gal: OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources Using Sequence Semantics. EDBT Workshops 2006: 573-576.
19. Jayant Madhavan, Philip A. Bernstein, Erhard Rahm: Generic Schema Matching with Cupid. VLDB 2001: 49-58.
20. Kewei Tu, Yong Yu: CMC: Combining Multiple Schema-Matching Strategies Based on Credibility Prediction. DASFAA 2005: 888-893.
21. Alexander Maedche, Boris Motik, Nuno Silva, Raphael Volz: MAFRA - A MApping FRAmework for Distributed Ontologies. EKAW 2002: 235-250.
22. V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics-Doklady*, Vol. 10, pages 707-710, August 1966.
23. http://daml.umbc.edu/ontologies/ittalks/person. Visit date: 28-jul-2008.
24. Ontology that describe data from a person. http://www.cs.umd.edu/projects/plus/DAML/onts/ personal1.0.daml. Visit date: 28-jul-2008.
25. Ontology Evaluation Initiative. http://oaei.ontologymatching.org. Visit date: 30-jul-2008.
26. Marc Ehrig: Ontology Alignment: Bridging the Semantic Gap. (Contents) 2007, Springer, ISBN 978-0-387-36501-5.
27. Marc Ehrig and York Sure. Ontology mapping - an integrated approach. In ChristophBussler, John Davis, Dieter Fensel, and Rudi Studer(Eds.): *Proceedings of the 1st ESWS*, LCNS 3053, pages 7691, Heraklion (GR), Springer-Verlag, 2004.
28. Natalya Noy and Mark Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. *Proceedings of AAAI-2000*, pages 450455, Austin (TX US). MIT Press/AAAI Press, 2000.
29. Natalya Noy and Mark Musen. Anchor-prompt: using non-local context for semantic matching. In *Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 63-70, 2001.
30. Anhai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos and, Alon Halevy. Learning to match ontologies on the Semantic Web. *The VLDB Journal*, **12**:303-319, 2003.
31. Marc Ehrig and Steffen Staab. QOM - Quick Ontology Mapping. *Proceedings of 3rd ISWC*, Hiroshima (JP), pages 683-697, Springer-Verlag, 2004.
32. http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/ontologies/russia1.owl. Last visit: 24-jul-2008.
33. http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/ontologies/russia2.owl. Last visit: 24-jul-2008.
34. Yi Li, Juan-Zi Li, Duo Zhang, Jie Tang: Result of Ontology Alignment with RiMOM at OAEI'06. Ontology Matching 2006.
35. Frank van Harmelen: Two Obvious Intuitions: Ontology-Mapping Needs Background Knowledge and Approximation. IAT 2006: 11.

36. Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich: Discovering Missing Background Knowledge in Ontology Matching. ECAI 2006: 382-386.
37. Ruben Vazquez, Nik Swoboda: Combining the Semantic Web with the Web as Background Knowledge for Ontology Mapping. OTM Conferences (1) 2007: 814-831.
38. Rudi Cilibrasi, Paul M. B. Vitányi: The Google Similarity Distance. IEEE Trans. Knowl. Data Eng. 19(3): 370-383 (2007).
39. Risto Gligorov, Warner ten Kate, Zharko Aleksovski, Frank van Harmelen: Using Google distance to weight approximate ontology matches. WWW 2007: 767-776.
40. Marco Ernandes, Giovanni Angelini, Marco Gori: WebCrow: A Web-Based System for Crossword Solving. AAAI 2005: 1412-1417.
41. http://www.cs.yale.edu/ dvm/daml/bib-ont.daml. Visit date: 28-jul-2008.
42. http://www.isi.edu/webscripter/bibtex.o.daml. Visit date: 28-jul-2008.
43. AKT Ontology. http://www.aktors.org/ontology/portal. Visit date: 28-jul-2008.
44. http://www.cs.umd.edu/projects/plus/DAML/onts/cs1.0.daml. Visit date: 28-jul-2008.
45. http://orlando.drc.com/daml/Ontology/Genealogy/current/. Visit date: 28-jul-2008.
46. Ontology about the GEDCOM data model. http://www.daml.org/2001/01/gedcom/gedcom. Visit date: 28-jul-2008.
47. Carmel Domshlak, Avigdor Gal, Haggai Roitman: Rank Aggregation for Automatic Schema Matching. IEEE Trans. Knowl. Data Eng. 19(4): 538-553 (2007).
48. Qiu Ji, Weiru Liu, Guilin Qi, David A. Bell: LCS: A Linguistic Combination System for Ontology Matching. KSEM 2006: 176-189.
49. Jorge Martinez-Gil, Jose F. Aldana-Montes: Evaluation of two heuristic approaches to solve the ontology meta-matching problem. Knowl. Inf. Syst. 26(2): 225-247 (2011).
50. Jorge Martinez-Gil, Enrique Alba, José F. Aldana-Montes: Optimizing Ontology Alignments by Using Genetic Algorithms. NatuReS 2008.
51. J. Wang, Z. Ding, C. Jiang: GAOM: Genetic Algorithm based Ontology Matching In Proceedings of APSCC, 2006.
52. Gerard D. Cohen, Simon Litsyn, Gilles Zémor: On greedy algorithms in coding theory. IEEE Transactions on Information Theory 42(6): 2053-2057 (1996).
53. Yoon. Lee, Mayssam Sayyadian, AnHai Doan, Arnon Rosenthal: eTuner: tuning schema matching software using synthetic scenarios. VLDB J. 16(1): 97-122 (2007).
54. Pat Langley: Elements of Machine Learning. 1994, ISBN 1-55860-301-8.
55. Marc Ehrig, Steffen Staab, York Sure: Bootstrapping Ontology Alignment Methods with APFEL. International Semantic Web Conference 2005: 186-200.
56. Michael I. Jordan, Christopher M. Bishop: Neural Networks. The Computer Science and Engineering Handbook 1997: 536-556.
57. Jingshan Huang, Jiangbo Dang, José M. Vidal, and Michael N. Huhns. Ontology Matching Using an Artificial Neural Network to Learn Weights. IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition 2007.
58. Esko Ukkonen: Approximate String Matching with q-grams and Maximal Matches. Theor. Comput. Sci. 92(1): 191-211 (1992).
59. Giorgos Stoilos, Giorgos B. Stamou, Stefanos D. Kollias: A String Metric for Ontology Alignment. International Semantic Web Conference 2005: 624-637
60. Jorge Martinez-Gil, Ismael Navas-Delgado, Antonio Polo-Marquez, Jose F. Aldana-Montes: Comparison of Textual Renderings of Ontologies for Improving Their Alignment. CISIS 2008: 871-876.

61. Ricardo A. Baeza-Yates, Berthier A. Ribeiro-Neto: Modern Information Retrieval. ACM Press / Addison-Wesley 1999, ISBN 0-201-39829-X.
62. Michael K. Buckland, Fredric C. Gey: The Relationship between Recall and Precision. JASIS 45(1): 12-19 (1994).
63. Gerd Stumme, Alexander Maedche: FCA-MERGE: Bottom-Up Merging of Ontologies. IJCAI 2001: 225-234.
64. Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich: S-Match: an Algorithm and an Implementation of Semantic Matching. ESWS 2004: 61-75.
65. Wei Hu, Gong Cheng, Dongdong Zheng, Xinyu Zhong, Yuzhong Qu: The Results of Falcon-AO in the OAEI 2006 Campaign. Ontology Matching 2006.
66. Slawomir Niedbala: OWL-CtxMatch in the OAEI 2006 Alignment Contest. Ontology Matching 2006.
67. Patrick Lambrix, He Tan: A Tool for Evaluating Ontology Alignment Strategies. J. Data Semantics 8: 182-202 (2007).
68. Gerard Salton, Chris Buckley: Improving retrieval performance by relevance feedback. JASIS 41(4):288-297 (1990).
69. Alexandros Chortaras, Giorgos B. Stamou, Andreas Stafylopatis: Learning Ontology Alignments Using Recursive Neural Networks. ICANN (2) 2005: 811-816.
70. Jingshan Huang, Jiangbo Dang, José M. Vidal, and Michael N. Huhns. Ontology Matching Using an Artificial Neural Network to Learn Weights. IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition 2007.
71. Malgorzata Mochol, Elena Paslaru Bontas Simperl: A High-Level Architecture of a Metadata-based Ontology Matching Framework. DEXA Workshops 2006: 354-358.