Universidad Politécnica de Valencia

Departamento de Sistemas Informáticos y Computación

Máster de Ingeniería de Software, Métodos Formales y Sistemas de Información

Tesina de Máster

RECUPERACIÓN DE INFORMACIÓN BASADA EN DISTANCIAS HIPERSINTÁCTICAS

Autor: Héctor Valero Llinares

Director: Josep Francesc Silva Galiana

Índice

RESUMEN

La búsqueda y recuperación de información en Internet es una tarea difícil, debido a esto no es muy grande la cantidad de herramientas capaces de extraer información de páginas web en tiempo real. La principal causa es que la mayoría de las páginas web en Internet están implementadas usando (X)HTML "plano", el cual es un lenguaje que carece de información semántica estructurada.

Por esta razón gran parte de los esfuerzos en esta área se han encaminado a desarrollar técnicas de extracción de URL's. Este campo ha producido muy buenos resultados implementados por los motores de búsqueda modernos como Google[1], Yahoo[2] o Bing[3]. Por el contrario, la extracción de información de una única página web no ha conseguido tan buenos resultados, como demuestra la escasez de herramientas dedicadas a dicha tarea. Recientemente hemos definido una técnica nueva para la recuperación de información de una página web (1) o de colecciones de páginas web interconectadas (2). Esta técnica se basa en distancias sintácticas para recuperar información. Esto permite que la técnica pueda trabajar con cualquier página web, y, además, recuperar información "online". La implementación, disponible en (3), y experimentos demuestran la utilidad de la técnica.

[1] www.google.com
[2] www.yahoo.com
[3] www.bing.com

1. Introducción

La búsqueda de información siempre ha sido una tarea complicada. Hasta hace pocas décadas había que tener paciencia y perseverancia para hallar la información que se necesitaba a base de buscar y leer manuscritos, y en muchos casos nunca llegaba a encontrarse la información buscada.

Internet ha cambiado enormemente la sociedad, el acceso a la información ha vivido la mayor revolución de la historia, dando lugar al problema opuesto, la sobre-información, ahora el problema es que en Internet hay millones y millones de páginas web con muchísimos contenidos, encontrar la información oportuna vuelve a requerir paciencia y perseverancia.

Por ello, desde hace tiempo, *Information Retrieval (IR),* es una de las disciplinas de mayor interés en el mundo de la web y la web semántica. La escasez de herramientas online capaces de extraer información de manera automática y en tiempo real de páginas web nos muestra la dificultad que ello conlleva.

Las técnicas actuales de IR principalmente se basan en la recuperación de páginas web relacionadas con una consulta determinada (véase un estudio sobre esta técnica en (4)). En esta área, los motores de búsqueda como Google o Bing implementan algoritmos muy precisos y eficientes para la recuperación de páginas web relacionadas, pero, para ciertos propósitos, el nivel de granularidad de la información obtenida es demasiado grande, una página web entera, teniendo que buscar la información relevante para el usuario dentro de esa página.

En el caso de la web semántica, a menudo es posible producir resultados más precisos compuestos de textos que respondan a una pregunta dada. Sin embargo, esas técnicas suelen necesitar de un pre-procesado de las páginas web que van a ser consultadas. Se construye

un modelo ontológico y el conocimiento es modelado y consultado usando lenguajes como RDF (5) u OWL (6). Esto impone restricciones importantes a las páginas que pueden ser procesadas, además de que las herramientas implementadas suelen ser herramientas *offline.* Una de las razones es que la mayoría de las páginas web han sido implementadas con (X)HTML plano, el cual carece de información semántica estructurada. Y las herramientas que usan microformatos (7), (8) y (9) se enfrentan al mismo problema.

En este trabajo se introduce una técnica nueva para *IR* basada en distancias sintácticas. Informalmente hablando, la técnica busca un término especificado por el usuario y extrae de la página web aquellos elementos sintácticamente cercanos de este término. Por ello la técnica se apoya en la idea de que sintácticamente cercano implica semánticamente relacionado. Esta idea se extiende a distancias entre páginas y dominios usando distancias de hiperenlaces. Varios experimentos con nuestra implementación revelan que esta simple idea es bastante potente en la práctica. Las principales ventajas de la técnica es que no necesita usar proxies (10), puede trabajar *online* con cualquier página web sin necesidad de fases de pre-procesado (11), y puede recuperar información con un nivel de granularidad muy pequeño (una palabra, un párrafo, una línea de una tabla, etc.).

Las principales contribuciones de esta técnica se pueden resumir como:

1. Definición y formalización de distancia hipersintáctica.
2. Nuevos algoritmos para *IR* de múltiples páginas web.
3. La implementación de la técnica, la cual ha sido integrada en Firefox.
4. Un estudio empírico para medir el rendimiento de los algoritmos presentados.

2. Preliminares

Esta sección introduce unos conceptos básicos para una mayor comprensión del resto del trabajo. Se presentan unas definiciones previas y un algoritmo de recuperación de una única página web.

El DOM (Document Object Model) (12) es una API Estándar que permite acceder, añadir y cambiar dinámicamente contenido estructurado en documentos XML y HTML. Define la estructura lógica de los documentos y el modo en que se accede y manipulan.

El DOM es una estructura jerárquica (arborescente) donde unos objetos dependen de otros. Además, los objetos del DOM modelan desde la ventana del navegador hasta el elemento más pequeño contenido en una página web.

Ejemplo 1. En la Figura 1 se muestra un fragmento HTML correspondiente a una tabla muy simple, a la izquierda se puede observar su visualización, y a la derecha el código HTML que la genera. En la Figura 2 se muestra la representación (visual) del DOM correspondiente a la tabla anterior, siendo cada elemento HTML un nodo DOM. Notar como en la parte inferior de la Figura 2, los nodos de tipo <TD> tienen un atributo con el texto de la celda.

Celda 1	Celda 2
Celda 3	Celda 4

```
<table border="1px;">
<tbody>
<tr>
<td>Celda 1</td>
<td>Celda 2</td>
</tr>
<tr>
<td>Celda 3</td>
<td>Celda 4</td>
</tr>
</tbody>
</table>
```

Figura 1. Ejemplo de una tabla HTML.

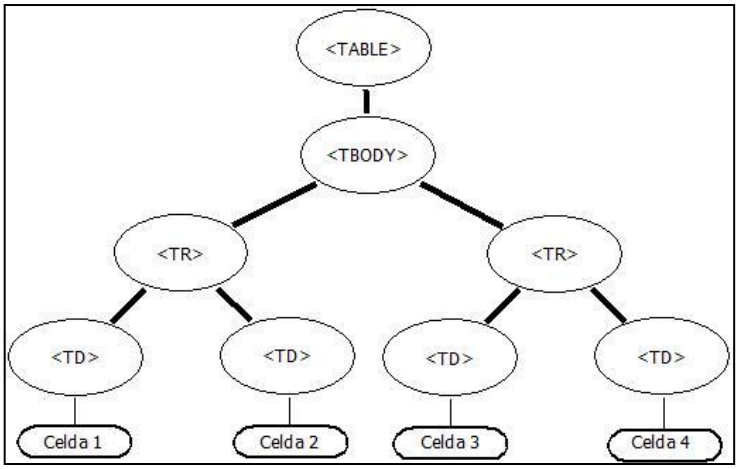


Figura 2. Fragmento de un árbol DOM.

Aunque cada nodo puede tener multitud de atributos, para simplificar la explicación de la técnica, en adelante se asumirá que cada nodo DOM tiene un único atributo de texto, evitando tener que entrar así en detalles de bajo nivel. Por supuesto la técnica revista los atributos oportunos dependiendo del tipo de nodo que se esté analizando, por ejemplo, si el nodo es de tipo hiperenlace es interesante buscar en el atributo *href*, atributo que está vacío en nodos de otro tipo.

Definición 1. Árbol DOM.

Sean:

$V = \{V_0, V_1, \dots, V_k\}$ *con* $k \in \mathbb{N}$ *un conjunto finito de vértices, siendo* V_0 *el vértice raíz y* $l(V_i)$ *la etiqueta del vértice* V_i.

$E = \{E_0, E_1, \dots, E_{k'}\}$ *con* $k' \in \mathbb{N}$ *un conjunto finito de arcos, de tal manera que* $\forall\, V_i$ *con* $1 \leq i \leq k$ $\exists!\, E_j$ *con* $0 \leq j \leq k'$ *tal que* $E_j = (V_{i'}, V_i)$.

Es decir, para cada vértice excepto el raíz, hay un solo arco que llega hasta él saliendo desde otro vértice. Con esto, se define un árbol DOM t = (V,E) como un árbol cuyos vértices V son nodos etiquetados con elementos HTML y están conectados por un conjunto de arcos E.

A menudo se referirá a la raíz de un árbol DOM *t* como *root(t).* Además se usará la notación $n \rightarrow^x n'$ para denotar que existe un camino de igual o menor tamaño que *x* entre los nodos *n* y *n'*.

Definición 2. Página web.

Sean:

t = (V,E) un árbol DOM

u = una URL

Se define una página web P como el par P = (u,t)

Las consultas del usuarios a menudo pueden contener múltiples palabras y metadatos, como el uso de las comillas (“ “) para búsqueda exacta, y operadores booleanos (*and, or, not*) para producir combinaciones complejas, o forzar la inexistencia de un determinado (sub)texto, etc. Sin embargo, para simplificar la formalización se asumirá que las consultas estarán compuestas por una única palabra. La extensión de la técnica para múltiples palabras es trivial (véase (1) para más detalles) y solo requiere la iteración del método sobre las palabras de la consulta. Para más detalles de implementación el lector puede consultar el código fuente de la implementación disponible públicamente en: http://users.dsic.upv.es/~jsilva/webfiltering.

Definición 3. Consulta.

Una consulta Q es un par Q = (w,d) donde

w es una palabra que está asociada con la información que es relevante para el usuario.

$d \in \mathbb{N}$ representa la tolerancia requerida en la búsqueda.

Definición 4. Resultado de una consulta.

Sean:

t = (V,E) un árbol DOM.

P = (u,t) una página web.

Q = (w,d) una consulta.

Se define el resultado de una consulta R = $\{t_0, \ldots, t_n\}$ como el conjunto de todos los sub-árboles de t tales que $\forall\ t_i = (V_i,E_i)$ con $0 \leq i \leq n$ se cumple:

- $\exists\ v \in V_i$ tal que $w \in l(v)$
- $\forall\ v' \in V_i$ se cumple que $v' \rightarrow^d v$

2.1 Recuperación de información de una única página web

El Algoritmo 1, explicado en detalle en (1), implementa un método para recoger información de una única página web que tomaremos como base para nuestros desarrollos. Claramente el algoritmo tiene un coste computacional lineal con el tamaño del árbol DOM.

En esencia, se buscan los *nodos clave*, que son los nodos cuya etiqueta contenga la palabra buscada. Desde esos nodos, el conjunto *nodos relevantes* se calcula como aquellos nodos cuya distancia sintáctica a algún nodo clave sea igual o inferior a la tolerancia especificada en la consulta.

Todos los *nodos ancestros* y *sucesores* de los nodos relevantes forman los nodos finales del árbol DOM filtrado, y el conjunto final de arcos es el que se deduce del conjunto final de nodos. Por lo tanto, la página web final (a la que se denominará en lo siguiente como *slice*[4]) será siempre una porción de la página web original, y esta porción mantendrá la estructura original de la información puesto que los caminos entre los elementos recuperados se mantienen.

Algoritmo 1. Recuperación de información de una página web

Entrada: Una página web $P = (u,t)$ y una consulta $q = (w,d)$
Salida: Una página web $P' = (u,t')$
Inicialización: $t = (v,e)$, $t' = (\phi, \phi)$

(1) $Key\_nodes = \{ n \in v \mid l(n) = w \}$

(2) $relevant\_nodes = \{ n \in v \mid n \rightarrow^{d} n' \wedge n' \in key\_nodes \}$

(3) $ancestors = \{ n \in v \mid n_0 \rightarrow^{*} n \rightarrow^{*} n_1 \wedge n_0 = root(t) \wedge n_1 \in relevants\_nodes \}$

(4) $successors = \{ n \in v \mid n_0 \rightarrow^{*} n \wedge n_0 \in relevants\_nodes \}$

(5) $edges = \{ (n,n') \in e \mid n, n' \in (successors \cup ancestors) \}$

return $P' = (u, (successors \cup ancestors, edges))$

[4] El término *slice* que significa "fragmento" proviene de la técnica *program slicing*. Mantenemos el anglicismo en esta tesis por ser un término muy utilizado en este área.

Ejemplo 2. Considérese la página web de la Figura 3, junto con su versión filtrada tras la consulta del usuario *("members",0).* Esta página web es producida por un árbol DOM enorme, el fragmento de este árbol que produce la parte señalada con un rectángulo rojo (parte derecha) se muestra en la Figura 4.

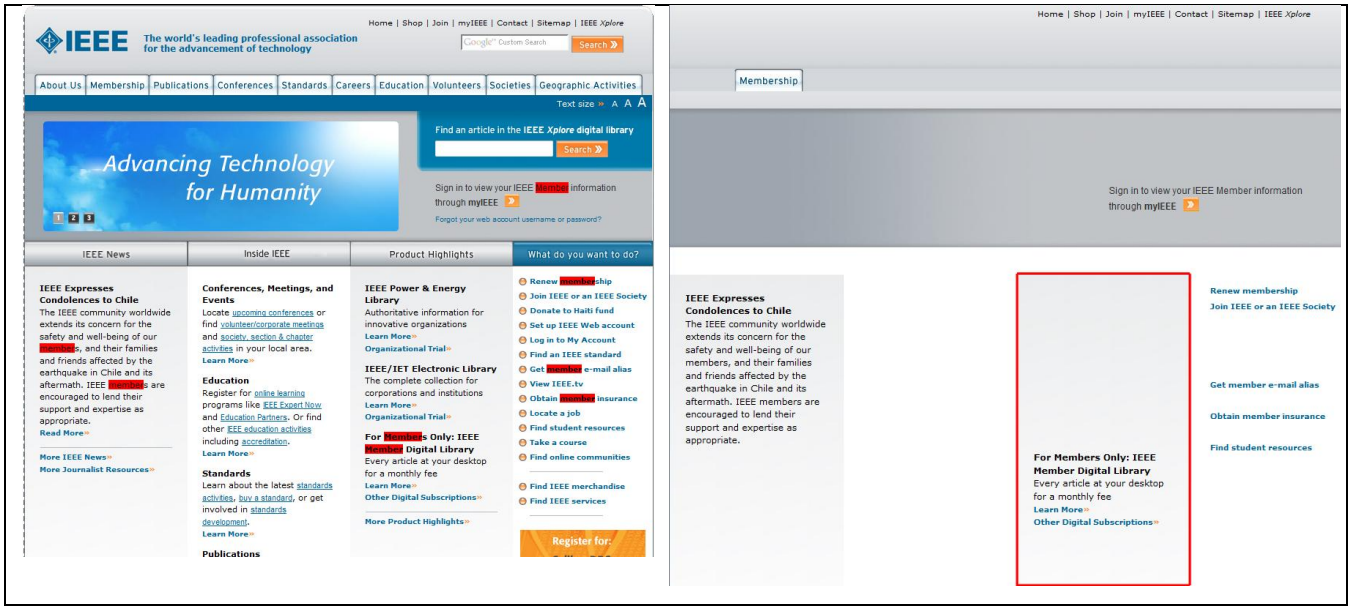


Figura 3. Página web principal de la IEEE (izqda.) y su versión filtrada (dcha.)

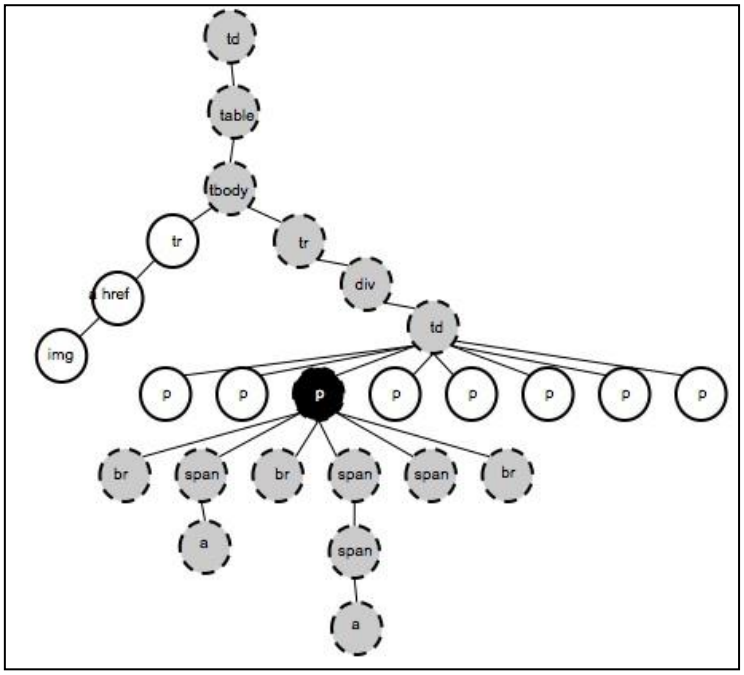


Figura 4. Sub-árbol DOM de la página web principal de la IEEE

En la Figura 4, el nodo negro es un nodo clave (i.e., contiene la palabra *members*). Como la tolerancia es 0, el nodo negro es el único nodo en el conjunto de nodos relevantes. El conjunto de nodos ancestros es el formado por los cinco nodos grises por encima del nodo negro, y el conjunto de nodos sucesores el formado por los nodos grises por debajo. Los nodos blancos no están relacionados con la consulta y son filtrados. Después de filtrar el árbol DOM con el Algoritmo 1, se obtiene la página web de la Figura 3 (derecha), donde sólo la información relacionada con la palabra “members” aparece.

3. Recuperación de información de múltiples páginas web

En esta sección se presenta la técnica desarrollada para la recuperación de información de un conjunto de páginas web interconectadas, pudiendo pertenecer estas páginas incluso a diferentes dominios. Se especifican varios conceptos para llegar finalmente a definir la distancia hipersintáctica, núcleo del algoritmo de recuperación de información de múltiples páginas web.

3.1 Motivación

La idea de este trabajo es modificar y extender la técnica vista en el apartado 2.1 para recuperar información de páginas web interconectadas mediante hiperenlaces. Desde una página web se quiere lanzar una consulta y que la técnica nos muestre la información relacionada con la consulta que haya en esa página web y aquellas a las que se pueda acceder (recursivamente) mediante hiperenlaces.

Todo esto conlleva una serie de problemas que hay que resolver para que la técnica pueda funcionar:

- *Espacio de búsqueda infinito*. Hay que definir una forma de elegir que hiperenlaces seguir.
- *Seguridad*. Al recuperar información de diferentes páginas web que además pueden llegar a ser incluso de diferentes dominios, hay que tener especial cuidado de que tipo de información se extrae (por ejemplo, código javascript)
- *Tiempo*. El tiempo en filtrar una única página web es muy pequeño, pero con un espacio de búsqueda potencialmente infinito es necesario definir una cota máxima de tiempo,

especialmente si se quiere que la técnica trabaje en tiempo real y online.

- El *retardo* de la carga de las páginas que la técnica decida explorar.
- La *presentación* de la información recuperada.

3.2 Recuperación de páginas web interconectadas

Esta sección extiende el Algoritmo 1 para la recuperación de páginas web interconectadas. En adelante se asumirá que el usuario ha cargado previamente una página web (que podemos llamar página web inicial) y que ha especificado un criterio de búsqueda para extraer información, de la página web inicial, de las que están interconectadas mediante hiperenlaces y de las páginas web que incluye (ej. frames o iframes). Se puede decir que todas esas páginas son las *páginas web interconectadas;* hay que observar que además no tienen porqué estar necesariamente en el mismo dominio

Tanto los frames como los iframes se pueden modelar considerando que sus árboles DOM son sub-árboles de la página web que los contiene. De esta manera, el Algoritmo 1 tiene la capacidad de extraer información relevante de páginas web compuestas estructuradas mediante frames.

Hasta ahora, para simplificar la explicación de la técnica, se había asumido que los nodos de un árbol DOM tenían un solo atributo o etiqueta. Para poder definir la conectividad entre páginas web necesitamos un atributo nuevo, es decir, los nodos DOM tendrán una etiqueta con un texto y otra etiqueta (que puede estar vacía) con un hiperenlace a otro nodo DOM, pudiendo pertenecer este nodo al mismo árbol (en el caso de las anclas) o a otro árbol DOM.

De esta manera, se pueden definir las nociones de páginas alcanzables y hiperespacio de búsqueda usados en el algoritmo de recuperación de información.

Definición 5. Alcanzabilidad.

Dada una página web P_0 se puede decir que una página web P_n es alcanzable desde P_0 (y lo denotaremos como $P_0 \rightarrow P_n$) sii $\exists\ P_0, P_1, \ldots, P_n \mid \forall\ P_i = (u, (V, E)), 0 \leq i \leq n-1, \exists\ v \in V / l(v) = u' \wedge P_{i+1} = (u', t)$.

Informalmente hablando, una página web es alcanzable desde otra página web si es posible seguir una secuencia de hiperenlaces que conecten ambas páginas.

Definición 6. Hiperespacio de búsqueda.

Dada una página web P = (u,t) se define el hiperespacio de búsqueda de P (H_p) como el conjunto de páginas alcanzables desde P:

$H_p = \{P_0, ..., P_n\}$ *tal que* $\forall P_i$ *con* $0 \leq i \leq n$ *se cumple que* $P \rightarrow P_i$

El espacio de búsqueda es la colección de páginas web que están relacionadas con la página web inicial, e (idealmente) deberían ser inspeccionadas por el algoritmo de recuperación de información. Sin embargo, el espacio de búsqueda de una página web a menudo es muy grande, y, además, es potencialmente infinito (más aún cuando se navega por páginas web dinámicas (13)). Por lo tanto se necesita reducirlo descartando algunos de los hiperenlaces. Además de todo esto, la técnica debe poder trabajar online. Esto implica que el tiempo de respuesta es un factor crítico, pero el análisis de una página web conlleva cargarla, lo cual es una tarea obligatoria que inevitablemente consume tiempo. Por lo tanto reducir el número de páginas web que deben ser analizadas (priorizando las más relevantes) es el principal objetivo de la técnica.

Con este propósito se define una distancia (denominada distancia hipersintáctica) entre nodos del espacio de búsqueda no necesariamente situados en la misma página web. La distancia es usada por la técnica para decidir qué nodos con hiperenlaces están más relacionados con la consulta especificada con el usuario y deberían ser explorados. Los demás se descartan, reduciendo así el espacio de búsqueda.

Usar distancias sintácticas para aproximar relaciones semánticas es una idea que ha sido demostrada mediante evaluaciones experimentales en diferentes trabajos. Por ejemplo, Micarelli y

Gasparetti (14) obtuvieron resultados empíricos que demostraban que las páginas web cuyos hiperenlaces iniciales están más próximos sintácticamente, están más relacionadas semánticamente entre sí que las páginas web cuyos hiperenlaces iniciales están más separados sintácticamente.

Para poder definir la distancia hipersintáctica se usarán los siguientes conceptos:

Definición 7. Distancia DOM (dT).

Sea t = (V,E) un árbol DOM y $V_i, V_j \in V$*, se define la distancia DOM (dT) entre dos nodos del mismo árbol DOM como:*

dT = el menor $n \in \mathbb{N}$ *de manera que* $V_i \rightarrow^n V_j$

Informalmente hablando, la distancia DOM entre dos nodos del mismo árbol DOM es la longitud del camino más corto entre ellos.

Definición 8. Distancia de Página (dP).

Sean P y P' dos páginas web pertenecientes al mismo dominio, se define la distancia entre páginas (dP) como:

dP = el menor $n \in \mathbb{N}$ *de manera que* $P=P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \cdots \rightarrow P_n=P'$

Se puede decir que la distancia de Página entre dos páginas del mismo dominio es el mínimo número de hiperenlaces que hay que atravesar para llegar de la primera página a la segunda

Definición 9. Distancia de Dominio (dD).

Sean $D=\{P_0, P_1, \dots, P_n\}$ y $D'=\{P'_0, P'_1, \dots, P'_m\}$ dos dominios diferentes (que se pueden ver como colecciones de páginas web). Se define la distancia de dominio (dD) como:

dD = el menor número de dominios diferentes que hay que atravesar de manera que $P \rightarrow P'$ con $P \in D$ y $P' \in D'$

Se usan la página inicial y los nodos clave como referencias para calcular las distancias. De esta manera, para un nodo dado, su distancia DOM es la longitud del camino entre este nodo y el nodo clave (*key_node)* más cercano en su árbol DOM; y las distancias de página y dominio se calculan con respecto a la página web inicial.

Definición 10. Distancia Hipersintáctica, Relevancia.

Sean $D=\{P_0,P_1, \dots, P_n\}$ y $D'=\{P'_0, P'_1, \dots, P'_m\}$ dos dominios, y $n \in P \in D$, $n' \in P' \in D'$ dos nodos DOM, se define la distancia hipersintáctica *de n con respecto a n' como:*

$$D = dT + K_P \cdot dP + K_D \cdot dD$$

donde K_P y K_D son constantes numéricas usadas para calibrar la ecuación. La Relevancia *R de un nodo DOM es la inversa de esta distancia hipersintáctica $R = 1 / D$. Las constantes K_P y K_D determinan la importancia que tiene el hecho de que una palabra especificada por el usuario esté en otra página o en otro dominio.*

Ejemplo 3. Considerar el espacio de búsqueda de la Figura 5:

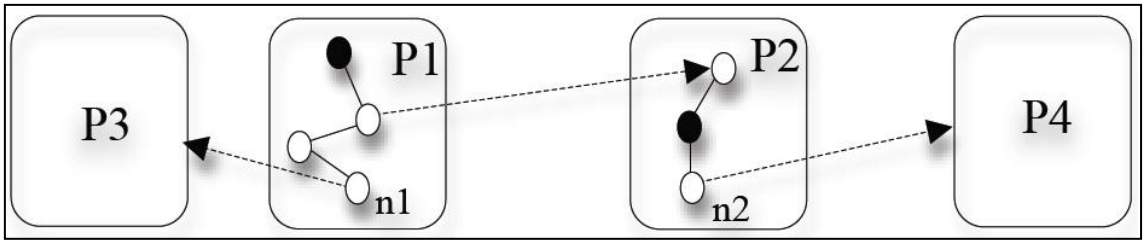


Figura 5. Ejemplo de un hiperespacio de búsqueda.

donde hay 2 nodos que contienen la palabra especificada por el usuario (los nodos coloreados de negro); el primer nodo está en la página web inicial (P1), y el segundo en la página web P2, y entre ellos hay una distancia de página de 1. Ahora, obsérvese que los nodos n1 y n2 son hiperenlaces hacia otras páginas web. La pregunta es: *¿Qué hiperenlace está más relacionado con la consulta del usuario y debería ser explorado primero por el algoritmo?* La respuesta está clara, el nodo más relevante y por ello el que tenga una menor distancia hipersintáctica. De acuerdo con la Definición 7, la relevancia depende fuertemente de los valores de las constantes K_P y K_D. Asumiendo que todas las páginas web están en el mismo dominio y que $K_P = 1$, tendríamos que D(n1) = 3 y D(n2) = 2, por lo que n2 sería más relevante. Por el contrario, si $K_P = 10$, obtendríamos que D(n1) = 3 y D(n2) = 11, siendo n1 más relevante que n2.

Después de muchos experimentos e intensas pruebas, se han seguido las siguientes decisiones de diseño:

1. Los hiperenlaces que se encuentran en la página web inicial son más relevantes que los que se encuentran en otra página web. Y lo mismo sucede a medida que la distancia de página se incrementa. Por lo tanto, la distancia DOM es más importante que la distancia de página.

2. Los hiperenlaces que se encuentran en el mismo dominio que la página web inicial son más relevantes que los que se encuentran en otro domino. Lo mismo pasa cuando la distancia de domino se

incrementa. Por lo tanto, la distancia de página es más importante que la distancia de dominio.

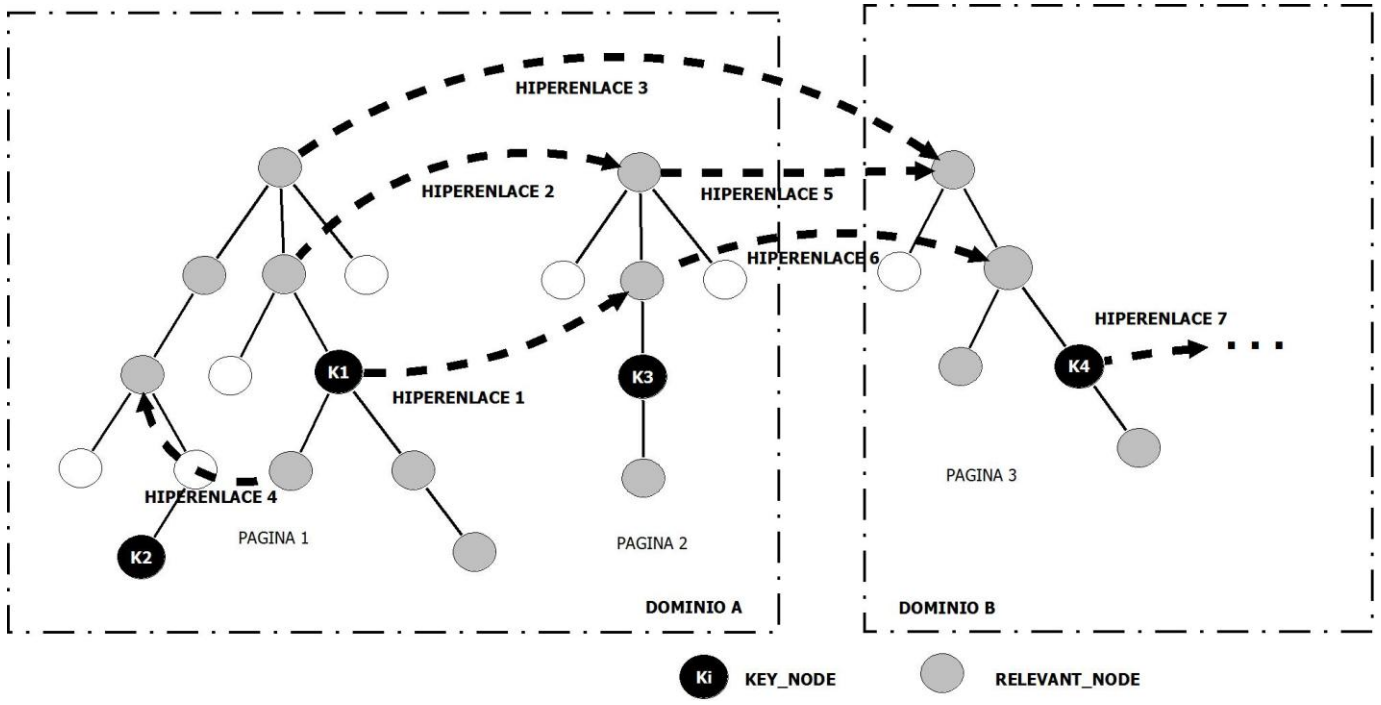
3. El algoritmo nunca debería analizar una página web con distancia de página superior a 5. Esta idea se apoya en estudios previos (Baeza y Castillo (13)) que demuestran que, en general, tres o cinco niveles de navegación son suficientes para llegar al 90% de la información que está contextualmente relacionada con la página web seleccionada por el usuario para la búsqueda.

De esta manera, considerando la cantidad de nodos que suele tener una página web, para cumplir con los 3 puntos anteriores, se ha decidido seguir los siguientes valores: $K_P = 10^6$ y $K_D = 10^9$. La cantidad de nodos que suele tener una página web suele ser menor que 10^3, una muestra de ello son los resultados obtenidos de nuestros experimentos (ver (3) para mas detalles) así, 10^6 asegura que la distancia entre dos páginas web diferentes sea siempre superior a la distancia entre dos nodos de la misma página. De la misma manera, la cantidad de páginas analizadas por nuestro método suele ser inferior que 10^2, con lo que 10^9 asegura que la distancia entre dos páginas web distintas de diferentes dominios sea siempre superior a la distancia entre 2 páginas web del mismo dominio. Por lo tanto definimos nuestra distancia hipersintácita como:

$$D = dT + 10^6 \cdot dP + 10^9 \cdot dD$$

Ejemplo 4. Considérese una página web inicial P1 y su espacio de búsqueda mostrado en la Figura 6. Se asume que el Algoritmo 1 ha analizado las 3 páginas web y que los nodos coloreados son los relevantes, los nodos clave son los negros, y los nodos blancos son los descartados. Para poder determinar qué hiperenlaces son más relevantes, se calcula la distancia hipersintáctica y la Relevancia de sus nodos DOM (véase la tabla). Esta información se usa para decidir qué hiperenlaces deben analizarse primero. Nótese en el ejemplo que la distancia hipersintáctica del nodo k4 es $0 + 1 * 10^6 + 1 * 10^9$. Este nodo tiene una menor Relevancia porque está en otro dominio. También se puede observar que el hiperenlace 4 no tiene distancia hipersintáctica;

esto es debido a que es un hiperenlace que apunta a la misma página web desde la que parte, estos hiperenlaces no hay que tenerlos en cuenta, pues el proceso previamente habrá filtrado la página completamente y ya habrá recogido toda la información relevante de dicha página, haciendo inútil un nuevo filtrado sobre ella. En la tabla se pueden observar las distancias hipersintácticas de cada hiperenlace.



Hiperenlace	dT	dP	dD	D	R
1	0	0	0	0	∞
2	1	0	0	1	1
3	2	0	0	2	0.5
4	-	-	-	-	0
5	1	1	0	$10^6 + 2$	$1 / (10^6 + 2)$
6	0	1	0	$10^6 + 1$	$1 / (10^6 + 1)$
7	0	1	1	$10^9 + 10^6$	$1 / (10^9 + 10^6)$

Figura 6. Información relevante extraída de diferentes páginas web y dominios.

En el árbol DOM de una página web se pueden distinguir entre hiperenlaces que pertenecen al slice e hiperenlaces que no pertenecen al slice. Aquellos hiperenlaces que no pertenecen al slice suelen estar relacionados con páginas que no tienen interés para el usuario, por consiguiente, para asegurar la calidad de la información recuperada, se decide que la cuarta decisión de diseño sea:

4. Los hiperenlaces que no pertenezcan al slice son descartados

En el ejemplo anterior ya han sido descartados los hiperenlaces que no pertenecían al slice, en concreto todos los hiperenlaces que salían desde nodos blancos.

Uno de los principales problemas de la extracción de información ocurre por la presencia de páginas web dinámicas: Una página web dinámica puede generar otra página web dinámica que contenga la palabra especificada por el usuario. Esta página web dinámica puede hacer lo mismo, y así sucesivamente sin fin. Esta situación se conoce como “agujero negro” porque los robots de búsqueda en esas páginas web tienen un espacio de búsqueda infinito donde siempre encuentran lo que buscan, quedando atrapados infinitamente si no se especifica algún tipo de límite en la búsqueda (13). Se puede observar que la combinación de las decisiones de diseño 3 y 4 evitan ese problema, pues la búsqueda se para en las páginas web que no contienen nodos y en las que su distancia de página es mayor que 5. Además de esto, hay una quinta decisión de diseño relacionada con el tiempo de respuesta de la técnica. Las reglas de usabilidad (15) establecen que 10 segundos es (como media) el tiempo límite que los usuarios esperan a que se cargue una página web. De esta manera se define otra decisión de diseño:

5. Se tiene que definir un tiempo de respuesta máximo, este tiempo será de 10 segundos siguiendo las reglas de usabilidad (15) a no ser que el usuario defina específicamente otra cantidad.

El tiempo usado para mostrar la información recuperada es constante, pero el tiempo de carga de páginas web es variable. Por ello la técnica utiliza un mecanismo para cargar las páginas web en orden

de Relevancia y extraer información de ellas. Cuando el tiempo haya llegado al límite o no queden hiperenlaces por explorar la técnica debe detener el análisis.

Algoritmo 2. Recuperación de información de múltiples páginas web

Entrada: Un conjunto de páginas web interconectadas con una página web inicial *P,* una consulta *q,* y un tiempo límite *T.*
Salida: Una página web *P'*
Inicialización: *paginaActual = P, linksPendientes = ϕ, tiempoInicial = getTime()*

do
relevant_nodes = getSlice(paginaActual, q)
show(relevant_nodes)
linksPendientes = linkPendientes U *getLinks(relevant_nodes)*
link = getMostRelevantLink(linksPendientes)
linksPendientes = linksPendientes / link
paginaActual = load(link)
while *not(timeout(T,tiempoInicial))*

return *P' (que se va mostrando incrementalmente por la función show)*

El Algoritmo 2 resume y explica de manera abstracta la técnica para la recuperación de información de páginas web interconectadas. Se usan las siguientes funciones que implementan las ideas y ecuaciones explicadas en esta sección:

- *getTime()*: Devuelve el instante de tiempo actual del reloj del sistema.

- *getSlice():* Obtiene el slice de una página web mediante el Algoritmo 1.

- *show():* Muestra en el navegador una colección de nodos DOM. Se debe mostrar de manera incremental. Su formalización no es el

objetivo de este trabajo, para más detalles del trabajo realizado en esta línea consultar (16).

- *timeout():* Comprueba si se ha cumplido algunas de las causas por las que el algoritmo debería detenerse (como haber superado el tiempo límite o que no queden hiperenlaces por procesar).

Algoritmo 2.1. Función timeout()

Entrada: El tiempo límite *T* del Algoritmo 2 y el instante de tiempo inicial *tiempoInicial*
Salida: Un valor booleano referenciando si el Algoritmo 2 debe detenerse o no
Inicialización: *resultado = FALSE*

tiempoActual = getTime()
if*(linksPendientes == ϕ)*
    resultado = TRUE
else if*(tiempoActual – tiempoInicial > T)*
    resultado = TRUE
return *resultado*

- *getLinks():* Extrae los nodos hiperenlaces de un conjunto de nodos. Esta función hace uso de *primerElemento()* que devuelve el primer elemento de la colección que se le pasa como parámetro.

Algoritmo 2.2. Función getLinks()

Entrada: Un conjunto de nodos *N*
Salida: Un conjunto de hiperenlaces *H*
Inicialización: *H = ϕ*

while *not(N== ϕ)*
(1) *nActual = primerElemento(N)*
(2) *N = N / nActual*
(3) ***if****(isLink(nActual)*
(4) *H = H* U *nActual*

return *H*

- *load():* Carga una página web. Se le debe pasar por parámetro la dirección de la página a cargar.

- *getMostRelevantLink() :* Calcula la distancia hipersintáctica de un conjunto de nodos para determinar cuál es el nodo más Relevante.

Algoritmo 2.3. Función getMostRelevantLink()

Entrada: Un conjunto de nodos *N*
Salida: El nodo más relevante
Inicialización: resultado = *primerElemento(N), menorDist = funcionDistancia(resultado)*

while *not(N== ϕ)*
(1) *nActual = primerElemento(N)*
(2) *N = N / nActual*
(3) ***if****(funcionDistancia(nActual) < menorDist)*
(4) *resultado = nActual*

return *resultado*

Nótese cómo se hace uso de la función *funcionDistancia,* a la que se le pasa como parámetro el nodo del cual se quiere obtener su distancia hipersintáctica tal como se ha definido en este trabajo. Para una mayor optimización de la implementación, durante la función *show()* (en la cual se muestra la información que la técnica va recuperando) se calculan las distancias hipersintácticas de los nodos con hiperenlace que se van mostrando, ahorrando cálculos posteriores.

Después de muchas pruebas empíricas y experimentos, se observó que el rendimiento de la herramienta mejoraba muchas veces si se incluyen todos los hiperenlaces de la primera página como hiperenlaces relevantes, dando siempre prioridad a aquellos que la herramienta había seleccionado con el Algoritmo 2. Esto es debido a que la mayoría de sitios web, en su página web inicial o página *index* no contienen información realmente, contienen un primer menú de hiperenlaces a otras páginas donde suele haber más información, por

ello se decidió tomar una última decisión de diseño que debe ser opcional para el usuario de la técnica, pues depende de la página inicial que elija para la búsqueda.

6. Todos los hiperenlaces de la página inicial son considerados como hiperenlaces explorables por la herramienta (independientemente de si forman parte o no del slice de la página inicial).

Nótese que esta decisión no afecta prácticamente al Algoritmo 2, la única modificación que habría que realizar es una segunda colección de hiperenlaces, que estaría compuesta por todos los hiperenlaces de la primera página menos los seleccionados por el algoritmo, esta colección inyectaría sus elementos en la colección linksPendientes cuando esta se quedara sin elementos. De esta manera se cumpliría el objetivo de esta decisión de diseño, que no es otra que aumentar el espacio de búsqueda siempre y cuando este espacio se haya visitado por completo.

Se debe observar también que, debido a esta decisión de diseño, el Algoritmo 2, en la inicialización, también debe llevar un parámetro que le indique si llevar a cabo la decisión de diseño 6 o no.

3.3 Ejemplo de un caso real

A continuación se muestra cómo sería un ejemplo real de recuperación de información usando la técnica presentada en este trabajo. Para simplificar el ejemplo se asumirá que el usuario no está interesado en seguir todos los hiperenlaces de la primera página (decisión de diseño 6), y solamente se descenderá un nivel de profundidad, es decir, el espacio de búsqueda es el conjunto de las páginas web formado por la página inicial y todas aquellas a las que se pueda llegar mediante un hiperenlace.

Considerar un usuario navegando por internet buscando información relacionada con "investigación" en la página web principal de la Universidad Politécnica de Valencia. Entonces el usuario busca hiperenlaces relacionados con investigación, y para cada hiperenlace que considere relevante, navegará a la nueva página web, leyendo su información y, otra vez, los hiperenlaces que considere relevantes para navegar de nuevo. Durante este proceso el usuario es forzado a (i) leer mucha información no relacionada con investigación que se muestra en las páginas webs visitadas y (ii) buscar los links relevantes. En un segundo escenario, el usuario utiliza nuestra herramienta para recuperación de información. En la página principal de la Universidad Politécnica de Valencia el usuario solamente tendrá que especificar que está interesado en la información relacionada a investigación. Un simple clic produce una página web nueva que contiene toda la información relevante de la página web principal y de sus páginas web hiperenlazadas relacionadas con investigación.

Figura 7. Página principal de la Universidad Politécnica de Valencia.

La Figura 7 muestra la página principal de la Universidad Politécnica de Valencia. Hay muy poca información relacionada con investigación. Nuestra técnica permite filtrar una página web y mostrar solamente la información relevante de acuerdo al criterio de filtrado. De esta manera, con la página de la Figura 7, el Algoritmo 1 produciría la nueva página filtrada que se muestra en la Figura 8.

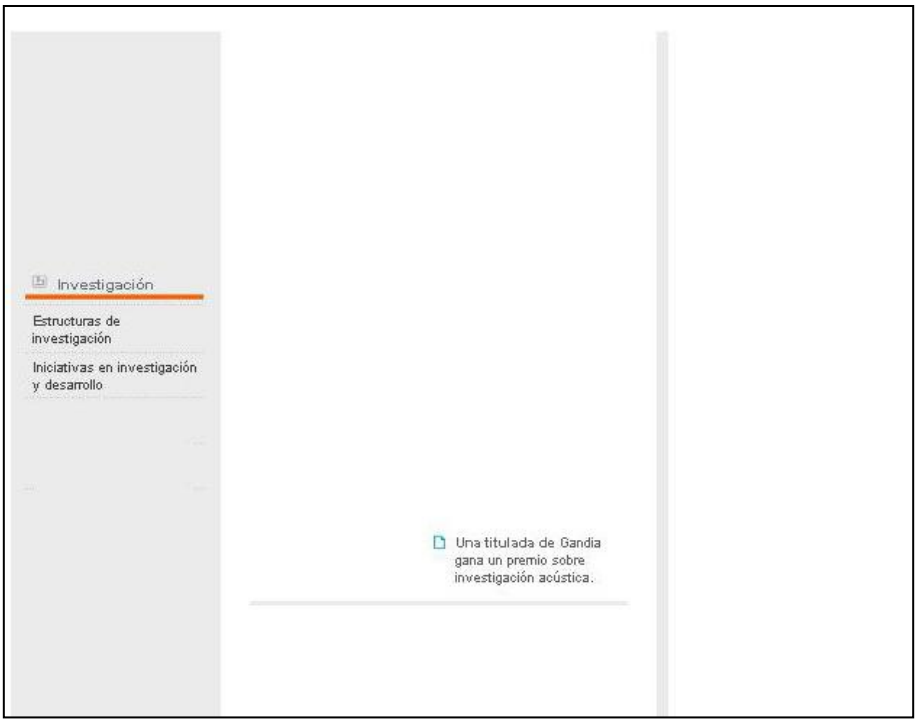


Figura 8. Slice de la página principal de la Universidad Politécnica de Valencia

Se puede observar que de todos los hiperenlaces de la primera página, la técnica tiene que decidir cuáles tiene que visitar, reduciendo todos los posibles hiperenlaces de la página inicial a solamente tres.

La palabra en la posición más alta (Investigación) en la Figura 8 solamente es una etiqueta, debajo hay dos hiperenlaces, cuyas páginas web destino se muestran en las Figuras 9 y 10; y en el centro, en la parte inferior de la imagen hay otro hiperenlace, cuya página web destino es la mostrada en la Figura 11.

Figura 9. Página web y apariciones de la palabra investigación

Figura 10. Página web y apariciones de la palabra investigación

El resultado de filtrar estas nuevas páginas web mostradas en las Figuras 9, 10 y 11 con el Algoritmo 1 es el que se muestra en las Figuras 11, 12 y 13 respectivamente. Se puede observar como algunas imágenes son filtradas y otras no, esto es debido a la existencia de la palabra investigación en los atributos de dichas imágenes.

Figura 11. Página web y apariciones de la palabra investigación

Figura 12. Slice de la página web de la Figura 7

Figura 13. Slice de la página web de la Figura 8

Figura 14. Slice de la página web de la Figura 9

Finalmente la técnica debe presentar el resultado de filtrar todas las páginas web visitadas durante el proceso en una única página web y de manera legible para el usuario. Uno de los objetivos de esta técnica es producir y mostrar al usuario los resultados en el menor tiempo posible. Además, como procesar una página web (filtrar la información relevante y mostrarla) es una tarea muy rápida, y la carga de páginas una tarea muy lenta, la visualización debe ser incremental e iterativa. Así se le puede ofrecer al usuario desde el primer momento información relevante que puede ir consultando mientras la técnica sigue visitando páginas web y recuperando información.

En la figura 13 se puede observar un ejemplo de visualización de la información recuperada en el ejemplo de esta sección. En ella aparece toda la información (una vez filtrada con respecto a investigación) de todas las páginas que la técnica ha visitado (con la premisa inicial de visitar sólo un nivel de profundidad para simplificar el ejemplo).

Figura 15. Resultado final de la recuperación de información

3.4 Implementación y experimentos

La técnica presentada ha sido implementada y enviada a Firefox, después de varias rondas de revisión, ha sido aceptado como plugin oficial de Firefox. La implementación permite al programador parametrizar la técnica para ajustar la cantidad de información recuperada, el número de páginas web exploradas, el modelo de visualización, y otras funcionalidades. Para poder determinar la configuración inicial por defecto, la herramienta fue inicialmente testeada con una colección de páginas web reales produciendo diferentes resultados y permitiendo ajustar los parámetros. Después se realizaron varios experimentos con páginas web reales y online. Concretamente se seleccionaron dominios con diseños y estructuras de página diferentes con el fin de estudiar el rendimiento de la técnica en diferentes contextos (páginas de noticias, fórums, páginas de organizaciones oficiales, etc.).

La tarea de evaluación empírica de esta técnica ha sido larga y tediosa debido a la carencia de benchmarks o bases de datos para poder evaluarla. Para la recuperación de páginas web enteras relacionadas con consultas específicas sí que existen benchmarks (17). El problema está en que el nivel de granularidad es demasiado grande para evaluar esta técnica, pues trabaja a nivel de páginas web enteras, y esta técnica trabaja con nodos, o conjuntos de nodos, es decir, de slices de páginas web. Estos benchmarks dicen si una página web de la base de datos está relacionada con una determinada consulta, pero una página web suele tener cientos o miles de nodos, siendo insuficiente este dato para evaluar esta técnica.

Por otro lado, también existen otros benchmarks que trabajan con un nivel de granularidad algo menor, especifican que parte(s) de una página web contiene(n) información relevante, pero no con respecto a alguna consulta, (como se hace en (18)), sino seleccionando la parte importante de cada página, un buen objetivo, por ejemplo, si se busca cambiar la visualización de las páginas web para adaptarla a dispositivos con pantallas reducidas, o recuperar la información relevante de cada página web y almacenarla, etc., pero tampoco sirve para evaluar esta técnica. Así que finalmente se tuvo que diseñar y

llevar a cabo cada experimento con el fin de evaluar correctamente la técnica presentada en este trabajo.

Esta técnica requiere de una página web inicial, desde donde iniciar el proceso de recolección de información y de un criterio de búsqueda, para realizar los experimentos se eligieron las páginas *index* o *home* de diferentes dominios.

Para cada dominio usado en las pruebas, se han diseñado dos experimentos diferentes. El primer experimento proporciona una medida del rendimiento medio de la técnica respecto a las medidas *recall, precision* y *F1* (ver en (19) una discusión sobre estas métricas). El segundo experimento mide el recall también, pero a nivel de páginas web enteras, qué, aunque no es el objetivo de esta técnica, nos ofrece otro punto de vista interesante de cara a posibles modificaciones para mejorar la técnica.

El objetivo del primer experimento era identificar la información que estuviera relacionada con una consulta particular (para cada dominio usado) del usuario. Para cada prueba, la herramienta exploraba varias páginas web siguiendo el Algoritmo 2 y extrayendo información relevante a la consulta del usuario, y se almacenaron los fragmentos de los árboles DOM que devolvía la herramienta. Luego, de cada página web visitada por la herramienta, se determinó el contenido que realmente era relevante de cada página web descargándola y seleccionándolo manualmente, tanto del contenido textual como del contenido multimedia, almacenando el árbol DOM de las partes seleccionadas para su posterior comparación.

La Tabla 1 resume los resultados obtenidos en el primer experimento. Para cada dominio, la primera columna contiene la **URL** de la página web inicial; la columna **Pages** muestra el número de páginas web exploradas por la herramienta en este experimento (el análisis se limitó a 10 segundos); la columna **Retrieved** contiene el número de nodos recuperados por la herramienta; la columna **Correct** corresponde al número de nodos recuperados que realmente eran relevantes respecto a la consulta; la columna **Missing** muestra el

número de nodos relevantes que no han sido recuperados por la herramienta; la columna **Recall** muestra el número de nodos relevantes recuperado por la herramienta dividido por el número de nodos relevantes totales (en las páginas analizadas para cada dominio); la columna **Precisión** muestra el número de nodos relevantes recuperados por la herramienta divididos por el número total de nodos recuperados por la misma; finalmente, la columna **F1** muestra la métrica F1 que se calcula como:

$$F1 = (2 \cdot P \cdot R) / (P + R)$$

siendo *P* la *precisión* y *R* el *recall.*

La primera conclusión importante de los experimentos es que, en 10 segundos, la herramienta es capaz de analizar 13,3 páginas de media . Y como la visualización es incremental, el primer resultado se muestra al usuario en menos de 1 segundo (10 / 13,3 segundos), evitando así una espera por parte del usuario para poder ir examinando los resultados.

Dominio	Pages	Retrieved	Correct	Missing	Recall	Precision	F1
www.ieee.org	10	4615	4594	68	98,74 %	99,54 %	99,03 %
www.upv.es	19	8618	8616	232	97,37 %	99,97 %	98,65 %
www.un.org	8	6344	6344	2191	74,32 %	100 %	85,26 %
www.esa.int	14	4860	4860	417	92,09 %	100 %	95,88 %
www.nasa.gov	16	12043	12008	730	94,26 %	99,70 %	96,90 %
www.mityc.es	14	12521	12381	124	99 %	98,88%	98,93 %
www.mozilla.org	7	6791	6791	14	99,79 %	100 %	99,89 %
www.edu.gva.es	28	10881	10856	995	91,60 %	99,79 %	95,51 %
www.unicef.es	9	5415	5415	260	95,41 %	100 %	97,65 %
www.ilo.org	14	1269	1269	544	69,99 %	100 %	82,34 %
www.mec.es	24	5527	5513	286	95,06 %	99,74 %	97,34 %
www.who.int	14	8605	8605	276	96,89 %	100 %	98,42 %
www.si.edu	18	26301	26269	144	99,45 %	99,87 %	99,65 %
www.sigmaxi.org	8	26482	26359	241	99,08 %	99,54 %	99,30 %
www.scientificamerican.com	7	5795	5737	97	98,33 %	98,99 %	98,65 %
ecir2011.dcu.ie	8	1659	1503	18	98,81 %	90,59 %	94,52 %
dsc.discovery.com	9	29097	29043	114	99,60 %	99,81 %	99,70 %
www.nationalgeographic.com	12	41624	33830	428	98,75 %	81,27 %	89,16 %
physicsworld.com	15	10249	10240	151	98,54 %	99,91 %	99,22 %

Tabla 1. Resultados de los Benchmarks

Los resultados muestran que la herramienta produce unos resultados altísimos de *recall* y *precisión.* La alta *precisión* de la herramienta no es una sorpresa, ya que la búsqueda sintáctica exacta asegura que la información recuperada esté, normalmente, muy relacionada con la consulta del usuario. Lo que sí que nos sorprendió más fue que el *recall* fuera tan alto, solo en pocos casos bajó del 75%. Y una de las causas de que en esos casos fuera menor era la existencia de sinónimos a la consulta del usuario que la herramienta actualmente ignora. La próxima versión incluirá un lexicón para resolver este problema.

En diez segundos los resultados son muy buenos debido a que la herramienta explora páginas web que están muy próximas a la página web inicial, y, en ese espacio de búsqueda es capaz de detectar con precisión las relaciones semánticas entre páginas.

Después de obtener estos buenos resultados, se han realizado más experimentos con el objetivo de comprobar si esta herramienta también podría ser usada en un proceso por lotes para recuperar información sin límite de tiempo, analizando todas las páginas web posibles. En este contexto, se quería conocer cuál era la cobertura de la herramienta a nivel de página. En (16), se diseña otro experimento en el que se recupera información de los mismos dominios que en el experimento anterior, permitiendo a la herramienta explorar tanto como le sea posible (es decir, las restricciones 3 y 5 son ignoradas). Se almacenó las *URLs* de las páginas web analizadas por la herramienta con respecto a la consulta del usuario.

Luego, con el *crawler* Nutch de Apache (20), se indexó cada dominio en su totalidad empezando desde la página web inicial y se almacenaron también las URLs de las páginas web que contenían la misma consulta del usuario, pudiendo así comparar los resultados.

En la tabla 2 se pueden observar los resultados de este experimento, la columna **URL** contiene la página web inicial de la búsqueda, la columna **Query** la consulta del usuario, en la columna **Toolbar** se muestra el número de páginas web recuperadas por nuestra herramienta y en la columna **Nutch** el número de páginas

recuperadas por el crawler de Apache, por último, en la columna **Recall** se muestra el recall obtenido.

A primera vista, parece que los resultados no son demasiado impresionantes, la herramienta explora, de media, el 30 % de todas las páginas web que realmente contenían la consulta del usuario. La causa de esto es que la técnica descarta automáticamente muchos hiperenlaces, concentrándose en el espacio de búsqueda más relevante; esto es debido a la restricción 4, que evita que la herramienta explore páginas web a las que se llegue desde páginas web que no contengan nodos relevantes. Suavizando la restricción 4 se permitiría a la herramienta explorar un espacio de búsqueda mayor, pero la precisión, probablemente, descendería significativamente.

Por otra parte, los experimentos se realizaron desde la misma máquina, y con la misma conexión de red, para intentar que las condiciones de las búsquedas fueran similares tanto con Nutch cómo con la herramienta, y si bien el recall de nuestra herramienta es bajo, el proceso de recolección de información tardó 10 segundos en completarse para cada búsqueda, frente a las varias horas que tardó Nutch en indexar cada dominio.

URL	Query	Toolbar	Nutch	Recall
www.ieee.org	competition	20	57	35.08%
www.upv.es	architecture	19	70	27.14%
www.un.org	violence	10	129	7.75%
www.esa.int	Venus	142	803	17.68%
www.nasa.gov	astronaut	144	527	27.32%
www.mityc.es	programa	43	106	40.56%
www.edu.gva.es	universitat	33	55	60%
www.unicef.es	Niños	12	87	13.79%
www.ilo.org	Child	121	755	16.02%
www.mec.es	estudiante	28	47	59.57%
www.who.int	Alcohol	14	31	45.16%
www.si.edu	Biology	8	238	3.3%
www.dsc.discovery.com	Tornado	165	246	67.07%
www.nationalgeographic.com	Projects	27	282	9.57%

Tabla 2. Resultados del segundo experimento

Toda la información relacionada con los experimentos, incluyendo el código fuente de la herramienta y más material se puede consultar en (3)

4.Trabajo futuro

Como se ha comentado anteriormente, esta técnica ha sido implementada como un plugin de Firefox en forma de barra (*toolbar)*, y, mientras se estudiaba la viabilidad de adaptarlo a otros navegadores y las repercusiones que esto tendría, surgió la idea de hacer la barra genérica, que funcionase independientemente del navegador y sistema operativo usados.

Para esto se está estudiando la posibilidad de que, en vez de ser una barra del navegador, utilizar un servidor donde alojar una página web, esta página web tendría dos frames o iframes, uno pequeño arriba, que contendría los mismos elementos que tiene actualmente la barra más una barra de direcciones, y, otro frame para la visualización de las páginas web.

De esta manera el usuario solo tendría que conectarse a dicha página, introducir la dirección de la página web inicial donde comenzar la búsqueda y especificar una consulta, obteniendo los resultados deseados independientemente del navegador y sistema operativo utilizado. En la Figura 16 se muestra un ejemplo de cómo podría ser la distribución de esta página web.

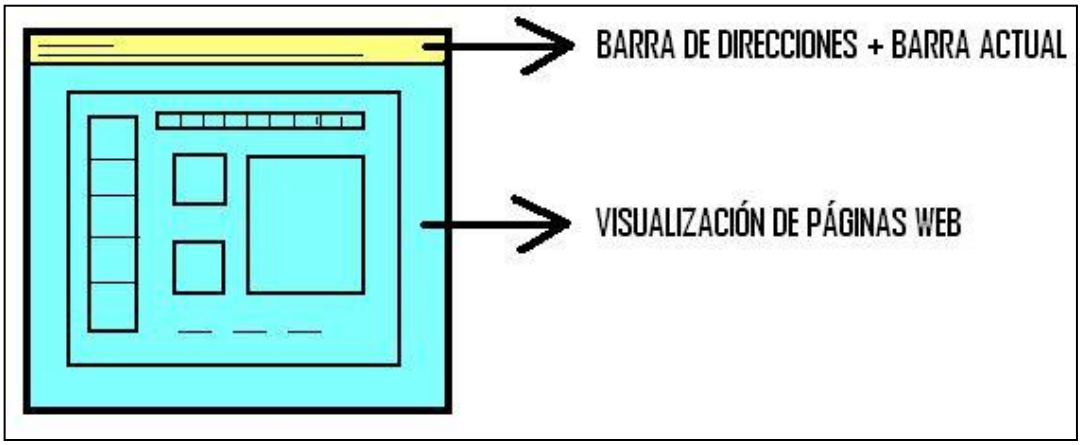


Figura 16. Página web con la barra incorporada

Pero para poder realizar todo esto se está estudiando qué cambios serían necesarios para poder llevarlo a cabo.

Por ejemplo, la interfaz XUL se podría cambiar por elementos HTML, pues ahora irían alojados dentro del frame superior de la página web.

El lenguaje de programación a utilizar tampoco tendría por qué suponer un cambio dramático para la implementación, de hecho, al ser un plugin de Firefox la lógica está implementada en Javascript, que es un lenguaje de scripts, pero, como norma general, los lenguajes de programación son más robustos y potentes que los lenguajes de scripts. Además, así la técnica se podría beneficiar de la programación multi-hilo, que, sin duda, es muy conveniente en herramientas de este tipo, donde varios hilos pueden estar cargando páginas y extrayendo información de ellas simultáneamente.

Hay que estudiar el problema de la seguridad, ya que aunque Firefox impone bastantes restricciones a la hora de programar un plugin, también es verdad que el propio navegador se ocupa de muchos temas de seguridad. Por ejemplo, el problema del XSS (Cross Site Scripting) y que básicamente consiste en no permitir que se ejecute código Javascript de terceros (ver (2) para más detalles).

Por otra parte, como ya se ha dicho anteriormente, se está estudiando el impacto de un lexicón, puesto que su uso ralentizaría el proceso, explorando menos páginas en el mismo tiempo. Hay que realizar multitud de experimentos empíricos para balancear entre el número de páginas obtenidas y la calidad de la información extraída.

También se está investigando como utilizar esta técnica para filtrar documentos XML , ya que se basa en el uso del DOM, el cual permite trabajar también con documentos XML, su extensión a documentos XML es un paso lógico.

5. Conclusiones

Este trabajo introduce una técnica de recuperación de información basada en distancias sintácticas. La técnica permite trabajar online y extraer información de sitios web sin ningún tipo de pre-procesado, etiquetado o indexado de las páginas web que se van a analizar.

Realmente es una técnica muy novedosa. Pues si ya es difícil la tarea de seleccionar qué páginas web están relacionadas con una consulta específica, realizar esta misma tarea, unido a la complicación de procesar estas páginas para quedarnos con los fragmentos o trocitos más relacionados con la consulta, y presentar los resultados de manera legible, da una idea de su dificultad.

Los experimentos han producido una medida *F1* del 96%, demostrando la utilidad de la técnica. El análisis de los resultados de los experimentos ha revelado que la existencia de sinónimos puede causar una pérdida de *recall*. Actualmente se está analizando el impacto de un lexicón. Usar sinónimos y relaciones semánticas permitirá aumentar la precisión del algoritmo, pero la eficiencia de la técnica se vería afectada. Se necesita una experimentación empírica para decidir si es mejor analizar muchas páginas web sin el uso del lexicón, o un número menor de páginas web utilizando el lexicón. Se debe estudiar el balance entre la cantidad y la calidad de la información recuperada

La implementación de la técnica ha sido integrada en la versión 1.5 de *Firefox WebFiltering Toolbar.* Esta herramienta es una extensión oficial que ha sido testeada y aprobada por Firefox.

Bibliografía

1. *Sergio López, Josep Silva: A New Information Filtering Method for WebPages. Seventh International Workshop on Text-based Information Retrieval - TIR 2010, 32 - 36, Bilbao, 2010.*

2. *Héctor Valero. Carlos Castillo. Josep Silva. Information Retrieval from Webpages Based on Syntax Disntances. 33rd European Conference on Information Retrieval. 2011.*

3. http://www.dsic.upv.es/~jsilva/webfiltering. [Online]

4. *M. Henzinger. The Past, Present and Future of Web Information Retrieval. Proceedings of the 23th ACM Symposium on Principles of Database Systems, 2004.*

5. *W3C Consortium. Resource Description Framework (RDF). www.w3.org/RDF.*

6. *W3C Consortium. Web Ontology Language (OWL). www.w3.org/2004/OWL.*

7. *Microformats.org. The Official Microformtas Site. http://microformats.org/, 2009.*

8. *R. Khare, T.Çelik. Microformats: a Pragmatic Path to the Semantic Web. Proceedings of the 15h International Conference on World Wide Web. Pages 865-866, 2006.*

9. *R.Khare. Microformats: The Next (Small) Thing on the Semantic Web? IEEE Internet Computing, 10(1):68-75, 2006.*

10. *Suhit Gupta et al. Automating Content Extraction of HTML Documents. World Wide Archive vol.8 issue.2, 179-224, 2005.*

11. *Po-Ching Li, Mind-Dao Liu, Ying-Dar Lin, Yuang-Cheng Lai. Accelerating Web Content Filtering by the Early Decision Algorithm. IEICE Transactions on Information and Systems vol. E91-D, pages 251-257. 2008.*

12. *W3C Consortium. Document Object Model (DOM). www.w3.org/DOM.*

13. *R. Baeza-Yaates, C. Castillo, Crawling the Infinite Web: Five Levels are eough. Lecture Notes in Computer Science, vol.2343, pages 156-157. Ed. Springer 2004.*

14. *A. Micarelli, F. Gasparetti, Adaptative Focused Crawling. The Adaptative Web, pages 231-262, 2007.*

15. *Jakob Nielsen. Designing Web Usability: The Practice of Simplicity: New Riders Publishing, Indianapolis ISBN 1-56205-810-X; 2010.*

16. *Carlos Castillo, Hector Valero, Josep Silva. Visualización de Información recuperada desde Múltiples Paginas Web. X Jornadas de Programación y Lenguajes. 2010.*

17. *LETOR: Learning to Rank for Information Retrieval. http://research.microsoft.com/en-us/um/beijing/projects/letor/.*

18. *Tim Weninger, William H. HSU, Jiawei Han. CETR: content extraction viat tag ratios. WWV '10 Proceedings of the 19th international conference on World Wide Web. 2010.*

19. *T. Gottron. Evaluating Content Extraction on HTML Documents. Proc. of the 2nd International Conference on Internet Technologies and Applications, pages 123-132, 2007.*

20. *Apache Foundation. The Apache crawler Nutch. http://nutch.apache.org, 2010.*