Mejora del Ranking de Búsquedas en un Contexto Legal

Juan Eduardo Roig, Héctor Oscar Nigro, Sandra González Císaro

INTIA- Departamento de Computación y Sistemas
Facultad de Ciencias Exactas - UNICEN
Campus Universitario - Paraje Arroyo Seco s/n
B7001BBO Tandil, Buenos Aires, ARGENTINA
TEL: +54-2293-439680 – FAX: +54-2293-439681
e-mail: jeroig@gmail.com, onigro@exa.unicen.edu.ar,
sagonci@exa.unicen.edu.ar

Abstract

In this paper, we develop and evaluate three probabilistic algorithms of user click-through behavior that are appropriate for modeling the click-through rate of legal documents. Our algorithms capture contextual factors related to the presentation as well as the underlying relevance for users. We focus on the positional context of the documents and the user's profile.
In this form we want to facilitate to the user the information that he looks for, showing to him documents that have been previously consulted by his colleagues and that probably are of interest for him.
A high yield library called Lucene is used for the development in the search of the legal documents. We test our models on real data obtained from studied juridical site during two months.

Key Words: Data Mining, Indexing, Information Retrieval, User Profile, Web Mining.

Resumen

En este artículo, desarrollamos y evaluamos tres algoritmos probabilísticos sobre el comportamiento de acceso de usuarios que son apropiados para modelar la tasa de acceso a un sitio de Internet que contiene documentos legales. Nuestros modelos capturan factores contextuales relacionados con la presentación así como la importancia subyacente para los usuarios. Nos concentramos en el contexto posicional de los documentos y en el perfil del usuario.
En esta forma queremos facilitar al usuario la información que él busca, demostrándole los documentos que han sido consultados previamente por sus colegas y que están probablemente de interés para él.
Para el desarrollo de la mejora en la búsqueda de los documentos legales, usamos una biblioteca de alto rendimiento llamada Lucene. Probamos nuestros algoritmos con los datos reales obtenidos durante dos meses del sitio jurídico en estudio.

Palabras claves: Indexación, Minería de Datos, Perfiles de usuarios, Recuperación de Información, Web Mining.

1 INTRODUCCIÓN

Durante el transcurso de los últimos, años Internet ha tenido una evolución constante tanto desde el punto de vista tecnológico como de usuarios conectados a ella. El avance vertiginoso de las tecnologías de la información y las comunicaciones (TIC), que se inició en el pasado siglo, matiza el mundo actual y a una sociedad que ha dado en llamarse sociedad de la información y del conocimiento, era de la información o era digital, entre otras denominaciones [1].
Internet ha propiciado la multiplicación de los canales y vías de acceso a la información y esto lleva a facilitar que millones de personas obtengan información para distintos fines: aprendizaje, entretenimiento, actualización, cultura general, etc., y lo mas asombroso, desde cualquier parte del mundo enlazada a la red.
Esta realidad también presenta un problema, el exceso de información que existe en la Web genera una dificultad real para los usuarios, ya que muchas veces no pueden encontrar lo que buscan [14], a raíz de esto nacen los motores de búsquedas. Un motor de búsqueda es un sistema informático que indexa un volumen de información para luego buscar sobre la misma.
En consecuencia, un motor de búsqueda en Internet almacena información de la WWW, para que luego los usuarios realicen sus búsquedas; las mismas se hacen con palabras clave o con árboles jerárquicos por temas, entregando como resultado un listado de documentos o direcciones Web en los que se mencionan temas relacionados con las palabras claves buscadas[3].
Nuestro objetivo es mejorar una herramienta de búsqueda de información y la forma de hacerlo es en base al comportamiento previo que han tenido los usuarios con la información que han consultado. El factor que se tendrá en cuenta será la tasa click-trough [7, 8] (CTR sus siglas en inglés) que mide el número de veces que alguien ha hecho click sobre un documento en relación al número de veces que se ha mostrado dicho documento. Para ello es necesario registrar las “huellas” de los usuarios. Es decir, que comportamiento han tenido, que documentación han consultado, que les ha sido de interés para ellos y que no. Una vez obtenida esta información proceder a plantear algoritmos para mejorar el ranking de documentos en las búsquedas. Una huella es un rastro o marca que queda registrada por parte del usuario al acceder a un determinado objeto. Las huellas se basan en la inclusión dentro de las páginas que se quieren controlar de una referencia (enlace) a un elemento adicional, que va a provocar una nueva petición por parte del cliente para acceder a ese elemento, con el efecto colateral de registrar el acceso a la página que la contiene[2, 10, 11, 13].

2 HERRAMIENTA DE RECUPERACIÓN DE INFORMACIÓN

Para el desarrollo de la mejora en la búsqueda de los documentos jurídicos, se toma como referencia una librería de alto rendimiento llamada Lucene; la cual es escalable basado en Recuperación de Información (IR) [5]. La misma permite la creación de índices y capacidad de búsquedas. Es un proyecto maduro, libre, de código abierto e implementado en Java; es miembro de la familia de proyectos de Apache Jakarta, autorizado bajo la licencia de Software libre de Apache. Lucene es actualmente, y ha sido durante años, la librería mas popular de Recuperación de Información en Java.
Proporciona un núcleo API (Interfaz de programación de aplicaciones) simple aunque poderoso, que requiere el entendimiento mínimo de incluir en un índice un texto completo y la búsqueda del mismo. Con aprender un conjunto mínimo de clases, ya se puede hacer uso de la misma.
Consiste en indexar texto para luego buscar en el mismo, y esto realmente lo hace bien, la librería puede adaptarse a cualquier dominio sin importar en que negocio se utilice, ocultando la complejidad de indexar y buscar detrás del uso de una simple API.

Lucene se puede ver como una capa, de la cual uno hace uso de la misma, independiente de la aplicación que tenga. Lucene nos permite generar índices y búsquedas en nuestro sitio de contexto legal y multilingüe [6]. Indexa documentos sobre los cuales luego se realizaran búsquedas, no se preocupa por la fuente de los datos, ni su formato, y ni siquiera su lengua, mientras se puedan convertir en texto.
Esto significa que podemos usarlo para indexar y buscar datos como páginas Web sobre servidores remotos, documentos almacenados de forma local, archivos de texto simples, documentos de Microsoft Word, HTML o archivos PDF, o cualquier otro formato del que previamente se puede extraer la información textual.
Los factores que se tienen en cuenta para el cálculo de una puntuación a un documento es la siguiente, figura 1. La cuenta es calculada para cada documento (d) [4, 14].

$$\sum_{t\ in\ q} tf\ (t\ in\ d) * idf(t) * boost(t.field\ in\ d) * lengthNorm(t.field\ in\ d)$$

Figura 1 Formula utilizada por Lucene para determinar la puntuación de un documento

El factor boost ("peso") es incorporado en la ecuación para poder afectar una pregunta o la influencia de campos sobre el calculo. El campo "boost" entra explícitamente en la ecuación como "t.field en d". El valor por defecto de este campo lógicamente es 1.0. Durante la creación de índices, a un documento se le puede asignar un "peso".
Tal factor dentro del documento implícitamente pone ese valor en los campos en que se especifique. El peso puede ser multiplicado por el valor de partida, dando un peso adicional al documento. Es posible agregar más de un peso distinto a cada documento, aunque para cada peso distinto, se debe crear un campo nuevo.

2.1. Asignando Pesos a Documentos

No todos los campos que se definen en un documento son creados iguales, incluso uno mismo puede generar nuevos campos y asegurarse de que tales campos tomen valores por algún criterio. El peso de un documento mediante un campo es un rasgo que tendremos en cuenta para cambiar el ranking de una búsqueda. Por defecto, todos los documentos no poseen boost (peso); o mejor dicho todos tienen el mismo peso de 1.0. Cambiando el peso de un documento, instruimos a Lucene de considerarlo más o menos importante respecto a otros documentos en el índice.

2.2 Algoritmos de Asignación de Pesos

2.2.1 Algoritmo Base CTR

El primer algoritmo que aplicaremos será la definición propia del CTR que es:

CTR = #veces clickeado / # veces visto

Con lo cual el peso (**boost**) de un documento **vid** viene dado por la siguiente formula:

Boost(vid) = CTR (vid) = #veces clikeado(vid) / #veces visto(vid)

Analizando la fórmula mencionada haremos una corrección a la misma ya que, supongamos que un documento con id='127005' fue visto 40 veces y clickeado 20 veces, lo cual, su peso seria de 0.5.

En este punto cabe mencionar que si el peso del documento es menor a uno (1) el mismo será penalizado, mientras mas se acerque a cero menor peso tendrá, y el efecto opuesto ocurrirá si el valor del documento es mayor a uno, concluyo, si el peso es uno no será ni penalizado ni beneficiado.
Entonces, usando la fórmula mencionada, acabaríamos penalizando, y en consecuencia quedarían mas abajo en los resultados, a todos los documentos relevantes.
Volviendo al ejemplo, si el CTR ('127005') es de 0.5, y el CTR medio de todos los resultados durante el historial es 0.2 (se pulsa un 20% de los documentos mostrados), el peso del documento, para este caso, seria entonces de **0.5 / 0.2 = 2.5**
Mediante este análisis, afirmamos que hay que comparar con el CTR global, quedando la fórmula de la siguiente manera:

Boost(vid) = CTR(vid) / CTR_medio

Donde:

CTR(vid) = #veces clikeado(vid) / #veces visto(vid)
CTR_medio = #Veces clickeado(todos documentos) / #veces visto(todos documentos)

Luego de anunciar tal fórmula realizaremos una pequeña, pero no menos importante aclaración.
Consideremos el siguiente ejemplo:

Con un CTR global de 0,5
Caso 1: el documento 'A' fue visto una vez y nunca clickeado.
Caso 2: el documento 'B' fue visto mil veces y nunca clickeado.
CTR(A) = 0 / 1 = 0 => **Boost(A) =** 0 / 0,5 **= 0**
CTR(B) = 0 / 1000 = 0 => **Boost(B) =** 0 / 0,5 **= 0**

Para el primer caso esta claro que, si bien este documento debe ser penalizado, no puede tener el mismo peso que otro documento que fue visto mas veces y nunca clickeado(caso 2), la lógica indica que el documento 'B' es, de momento, de menos importancia que el documento 'A', al menos hasta que 'A' no se halla mostrado mil veces y aun sigan sin seleccionarlo.
Se observa que nuestra fórmula no refleja estos tipos de casos, ya que para ambos documentos el peso es el mismo. En realidad la mejor estimación de probabilidad de que un usuario clickee en un documento es:

CTR_estimado(vid) = #veces clikeado(vid) + 1 / #veces visto(vid) + 2

Esta fórmula se llama Regla de Laplace[12]; y se basa el Teorema de Bayes asumiendo ningún conocimiento a priori.
Dado este refinamiento, veamos, para el ejemplo explicado con anterioridad, sus nuevos pesos:

CTR_estimado(A) = 0 + 1 / 1 + 2 = **0.33**
CTR_estimado(B) = 0 + 1 / 1000 + 2 =~ **0.001**

Donde

Boost(A) = 0,33 / 0,5 = **0,66**
Boost(B) = 0,001/ 0,5 = **0,002**

Concluimos que, la importancia de la corrección es poca si exigimos que el documento haya sido mostrado muchas veces (por ejemplo N=30); entonces la frecuencia y la probabilidad casi coinciden; aunque con N bajo su estimación no tiene tanta exactitud.
Para obtener el peso de un documento "vid" con el algoritmo base usaremos la siguiente fórmula:

Boost(vid) = CTR_estimado(vid) / CTR_medio

2.2.2 Algoritmo CTR mediante Heurística

Una supuesta mejora del algoritmo anterior seria, no solo incluir el CTR sino también tener en cuenta la posición en la que un documento es clickeado.
Para que lo veamos claramente, no es lo mismo un documento que fue clickeado veinte veces en la posición uno o dos, a que haya sido clickeado veinte veces en la posición quince, es más importante este último. Esto se debe, como mencionamos anteriormente, a que en una búsqueda los usuarios tienden a clickear en las primeras posiciones sin interesarse demasiado en estar seguros de que sean esos los documentos que necesiten.
Para este algoritmo tendremos en cuenta la posición en la que un documento es clickeado, con lo cual, es necesario la inclusión de una variable que se sumara al final de la ecuación, la misma será el "Valor Agregado" (VA) que tendrá cada documento cuando se calcule su CTR.
La fórmula es la siguiente:

Boost(vid) = (CTR_estimado(vid) / CTR_medio) + VA

Donde **VA = (Log(sum_posiciones_click) / Log (10)) / #veces_clickeado**, logrando de esta forma regularizar casos como los explicados anteriormente.
Tal algoritmo posee una condición, y es la de que si un documento nunca fue clickeado, su VA pasara a tener el valor cero.
Es importante que exista tal condición ya que sino el programa arrojaría un error de cálculo. Si un documento fue siempre visto pero jamás clickeado su *"suma de posiciones"* (en la que fue clickeado) seria cero y en consecuencia el **log(0) = error**.

2.2.3 Algoritmo CTR utilizando Optimización Multivariada

Este algoritmo es un modelo que tendrá en cuenta la posición del documento, no solo en donde se ha hecho click del mismo, sino también en que posiciones se ha observado sin haber sido seleccionado.
Al igual que en los algoritmos anteriores nos basamos en la formula **Boost(vid) = CTR_estimado(vid) / CTR_medio,** aunque extendemos la misma para tener en cuenta la posición e indicar la relevancia de la misma. Esto lo haremos mediante un caso de optimización multi-variada.
Para cada posición **"n"** definimos **boost_n** el que seria el peso (boost) si solo tuviéramos en cuenta los clicks y las visualizaciones que se han producido en esa posición:

boost_n = CTR_estimado_posicion_n / CTR_medio_posicion_n

Donde

CTR_estimado_posicion_n = (veces_clickeado_en_n + 1) / (veces_visto_en_n + 2)

En el caso ideal, los **boost_n** coincidirían para todas las **"n"** donde ha habido clicks, y este seria el valor que tomaríamos. Aunque generalmente nunca pasa esto, ya que, boost_1 es distinto a boost_2; boost_2 es distinto a boost_3 y así sucesivamente, con lo cual debemos tomar un valor boost "medio" que sea razonablemente bueno para todas las posiciones.
Para lograr tal propósito definimos una magnitud del error que cometemos separándonos del **boost_n** en la posición **n**:

$err\_n(b) = (boost\_n - b)^2$

El error global **"Err"** será la suma de los errores con un peso que representa la importancia de la posición **"n",** que definimos como la cantidad de clicks en esta posición:

$$\mathbf{Err(b) = \sum (err\_n(b) * v\_n)}$$

Con **v_n = veces_visto_en_n**

Planteado lo anterior, el próximo paso es encontrar el valor de **"b"** que minimiza **Err(b),** se trata de un problema simple de minimización que resolvemos con el método usual, derivando e igualando la derivada a cero.

$$\mathbf{Err\ (b)\ ' = 0}$$

Realizando el desarrollo de lo mencionado y aislando **"b"**, llamado de ahora en adelante **boost_factor**, obtenemos la siguiente formula:

$$\mathbf{b = boost\_factor = U / L}$$

Donde:

$$\mathbf{U = \sum (v\_n * u\_n\ ^2)}$$

$$\mathbf{L = \sum (v\_n * u\_n * l\_n)}$$

u_n=CTR_estimado_posicion_n=(veces_clickeado_en_n+1)/ (veces_visto_en_n + 2)

l_n = CTR_medio_posicion_n para todos los documentos.

El detalle del modelo probabilístico es correcto, pero lo importante aquí es que tiene similitud respecto al algoritmo planteado con anterioridad, pero en vez de usar la posición del documento para premiarlo, usamos la inversa de la probabilidad de que un documento sea clickeado allí.

2.3 Criterios para Determinar los Documentos Vistos por los Usuarios

Las huellas que hemos creado, nos darán la posibilidad de registrar las acciones que realizan los usuarios tanto en las búsquedas como en los documentos que se consultan.

Un factor importante en las búsquedas es el CTR, este será nuestro factor a la hora de determinar si un documento es importante o no para un usuario.

Esta claro que el CTR nos da un feedback implícito, al usuario en ningún momento se le pregunta explícitamente si el documento que esta observando le es de interés o no, para determinar esto observaremos los clicks que el usuario realiza en el sitio. El feedback implícito tiene la ventaja de que puede juntar un volumen grande de información en tiempos muy cortos, aunque como contrapartida es mas difícil de interpretar y en algunos casos potencialmente ruidoso [9].

La estrategia que seguiremos será la siguiente:

- Cuando un usuario dentro del sitio realiza una búsqueda, dependiendo la cantidad de resultados devueltos, se paginaran o no los mismos. La paginación mostrará diez resultados por página y los accesos a distintos números de página se mostraran al pie del listado devuelto.
- Cuando un usuario selecciona un documento proveniente de una búsqueda, al mostrar el contenido, también mostrara dos links, uno al documento anterior y otro al documento siguiente, estos serán los que aparecen en el orden de la búsqueda anteriormente realizada.
- Cuando un usuario realiza una búsqueda, los dos primeros resultados serán tomados como resultados observados, pero aun no clickeados, esto es considerado de esta forma ya que, por la estructura del sitio el usuario inevitablemente observa los documentos que se encuentran en la posición uno y dos respectivamente. En cambio el resto de los resultados asumiremos que por el momento no han sido considerados (esto quiere decir, que ni siquiera fueron vistos).

- Supongamos una lista de resultados devueltos, particionados cada diez documentos por efecto de la paginación (D1, D2, D3,…D10).
    - Si el usuario clickea D5, asumimos que [D1...D4] los ha observado pero no los ha elegido.
    - De similar manera, si en la misma lista el usuario clickea D7 y D3 asumimos que ambos resultados son más relevantes que D1, D2, D4, D5, D6 entonces concluimos que estos últimos han sido observados pero no elegidos.
    - Si el usuario pasa a la siguiente pagina de la cual esta observando sin clickear ningún documento, consideramos que de D1 a D10 ha observado a todos pero no ha elegido a ninguno.
    - Si el usuario, dentro de la misma búsqueda pasa a cualquier otra página que no es la siguiente, decimos que, solo ha observado a los dos primeros resultados de la página en la cual estaba, sin considerar al resto de la lista [D3...D10].
    - Si el usuario esta visualizando un documento proveniente de una búsqueda y pulsa el enlace “anterior documento” o “siguiente documento” se asume que el documento a visualizar será de interés para el mismo, aunque no se puede asumir nada de la lista que devuelve la búsqueda.

De esta forma queda expresado de que manera se almacenan las acciones de los clientes y en que criterios nos basaremos para generar feedback implícito de documentos que los usuarios consideran importantes.

3 RESULTADOS

La recolección de datos se realizó durante dos meses, se dividió en iteraciones en la cual se recalculaban los pesos de tales documentos cada semana y los mismo eran re-indexados. El sitio Web fue modificado para que, los usuarios una vez iniciada una sesión se les asignara de forma aleatoria uno de los tres algoritmos de ordenamientos planteados o el ordenamiento que no tiene en cuenta el peso del documento; entre otras variables de sesión también se almacenaba el grupo al que pertenecía el usuario.
De esta forma, cuando los usuarios realizaban búsquedas en el sitio, se consultaba al motor de búsquedas teniendo en cuenta el grupo y el algoritmo que tenia asignado en ese momento.
Con la información obtenida se construyó un webhouse, para el mantenimiento e historial de las huellas de los usuarios.
En base a lo mencionado anteriormente hemos obtenido las siguientes conclusiones:

- Cualquiera de los tres algoritmos de ranking de ordenamiento planteados es mejor que el ordenamiento que se tenía con anterioridad. Esto significa que, determinar si se muestra o no un documento no solo por la palabra que se esta buscando sino también por la conducta que tienen los usuarios, es una mejora sin dudas. (fig. 2)

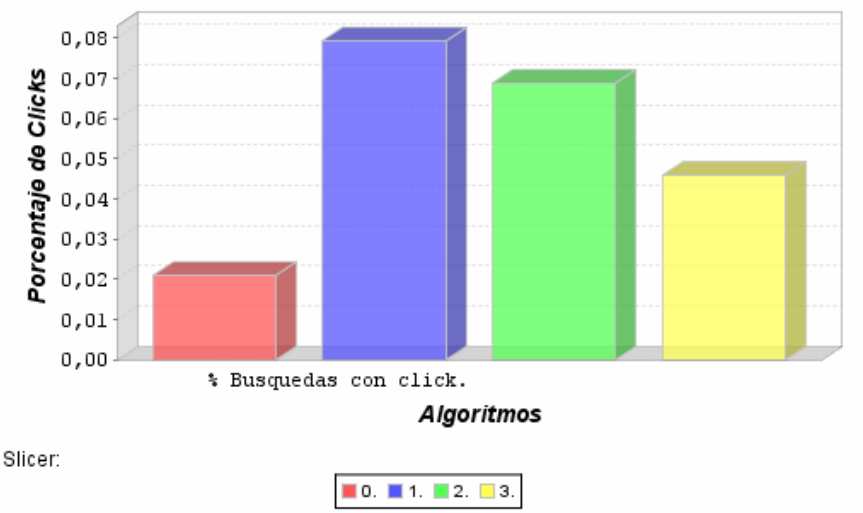

Figura 2: Porcentajes de clicks en búsquedas según algoritmos

- La cantidad de documentos observados en las búsquedas incrementa o decrecen en forma proporcional, sin tener en cuenta el ordenamiento. Podemos afirmar que los usuarios, independientemente de que se le mejore el ranking o no, van a observar los mismos documentos siempre. En la mayoría de los casos, los documentos de las dos primeras paginas. (fig. 3)

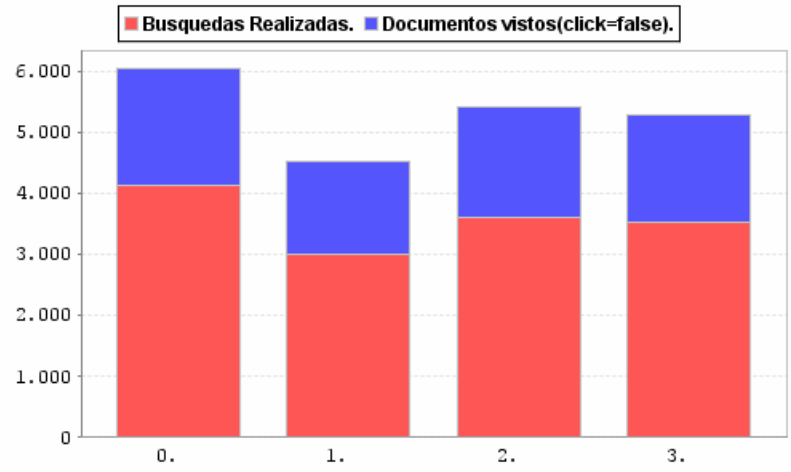

Figura 3: Documentos vistos según las búsquedas realizadas

- Se observa que en los algoritmos que se han clickeado más veces, se han realizado menos búsquedas. El hecho de minimizar búsquedas maximizando clicks nos dice que, cuando una búsqueda no devuelve resultados deseados en las primeras posiciones, se continúa con nuevas búsquedas hasta dar con el objetivo. (fig. 4)

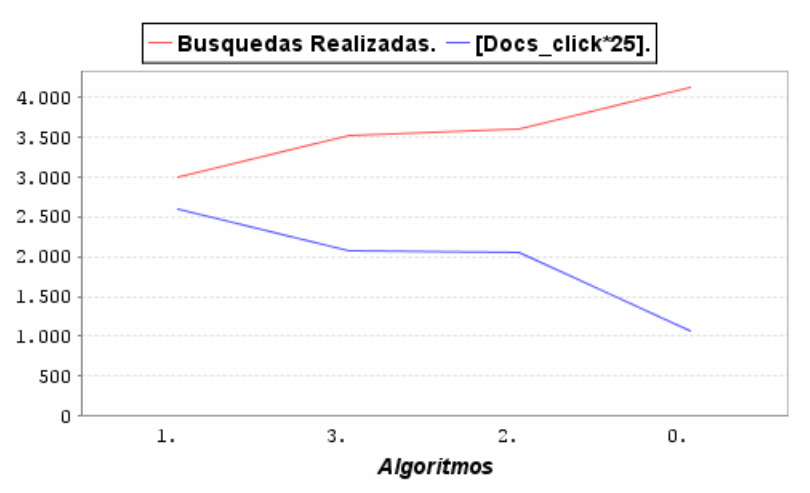

Figura 4: Documentos clickeados según las búsquedas realizadas

Los datos recopilados en el periodo de pruebas se basaron en un conjunto de documentos jurídicos, tales documentos son los más relevantes y en base a estos se han obtenido las conclusiones de este trabajo.
A lo largo de cada semana (iteración), no solo se iban almacenando los pesos de los documentos según el cluster y el algoritmo, también se recopilaban datos como cuantas veces fueron mostrados, cuantas veces clickeados, en que posiciones fueron seleccionados, además de la fecha en que se obtenían tales datos.
Esta información por un lado era necesaria para que cada semana se re-indexen los documentos con el nuevo peso calculado, pero también fue utilizada para ver resultados y comportamientos de los documentos.

4 CONCLUSIONES y FUTUROS TRABAJOS

En este trabajo se presentaron tres algoritmos de ordenamiento de resultados de documentos lanzada una búsqueda, tales ordenamientos ya no tienen en cuenta la visión clásica que puede tener cualquier motor de búsqueda, sino que consideran la importancia que tienen cada documento para los usuarios.
De esta forma se pretende facilitar al usuario la información que busca, mostrándole documentos que han sido consultados previamente por colegas suyos y que probablemente sean de interés para él.
Se pusieron a prueba los distintos ordenamientos, aunque a priori no se sabía cual seria el mejor. Se tomo un grupo de documentos más relevantes y se les realizo un seguimiento exhaustivo a los mismos.
Lo más importante de observar en los documentos es la variación del peso a lo largo de las iteraciones, ya que, este parámetro determinaría la convergencia o no de los algoritmos.
Tenemos un sistema “a priori” de ordenamiento de documentos y para ello asumimos, siendo esto clave; que si le damos un “peso” a un documento para que salga más arriba, eso disminuiría su CTR y al contrario, un peso negativo subirá su CTR. Dicho en fórmulas, una tesis central es que CTR (peso) es decreciente porque el CTR de un documento decaerá si subimos su “peso”.
Para ello, la fórmula **peso_n+1 = CTR_n / CTR_medio_n** (donde **“n”** es número de iteración) simplemente es una posible heurística de entre muchas pero no hay ninguna garantía de que este método sea convergente; podría darse la siguiente situación:

Iteración 1= peso 1; CTR/CTR_medio nos da = **1.4** (suponiendo)
Iteración 2 = ponemos el peso a 1.4; y entonces CTR/CTR_medio nos da **0.7**
Iteración 3 = ponemos el peso a 0.7, y entonces CTR/CTR_medio nos da **1.6**

Iteración 4 = peso 1.6; CTR/CTR_medio da **0.5**
etc...

Es decir, como nos pasamos en la corrección, en vez de ir al valor bueno, cada vez nos alejamos más: 1.4 => 0.7 => 1.6 => 0.5 => 1.7 => 0.3; es como si un mal conductor girara demasiado cada vez que quiere corregir la ruta por la que transita.

Nuestro OLAP de documentos nos muestra que los algoritmos aplicados tienden a converger ya que su peso por cada iteración no va teniendo picos ascendentes ni descendentes, sino que se regularizan buscando el peso real que debe tener el documento según el algoritmo aplicado. (fig. 5)

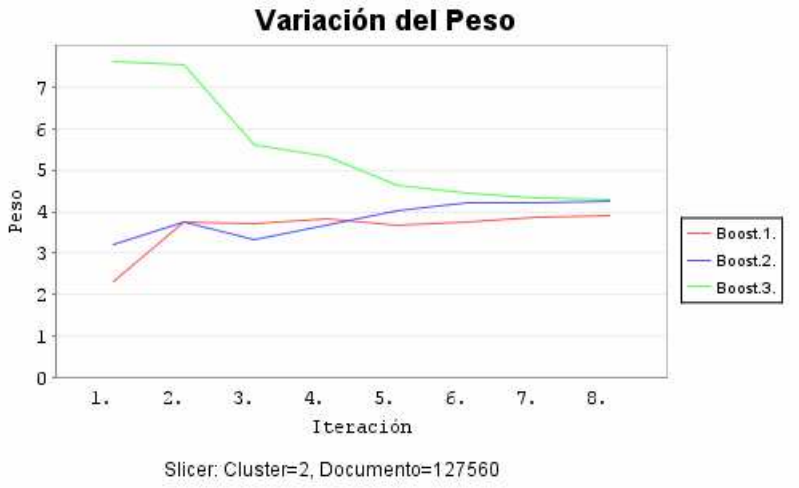

Figura 5: Pesos de un documento según algoritmos a través del tiempo

La figura 5 muestra la variación del peso del documento 127560 para el grupo de usuarios 2 respecto a cada semana; tal documento es el código civil y es el más visto dentro del sitio. Similar comportamiento han sufrido los demás documentos de los cuales se han almacenado datos de CTR.

El OLAP de documentos también nos muestra como a través de las iteraciones, la posición media de click va disminuyendo, es decir, clickean más cerca de la cima. (fig. 6)

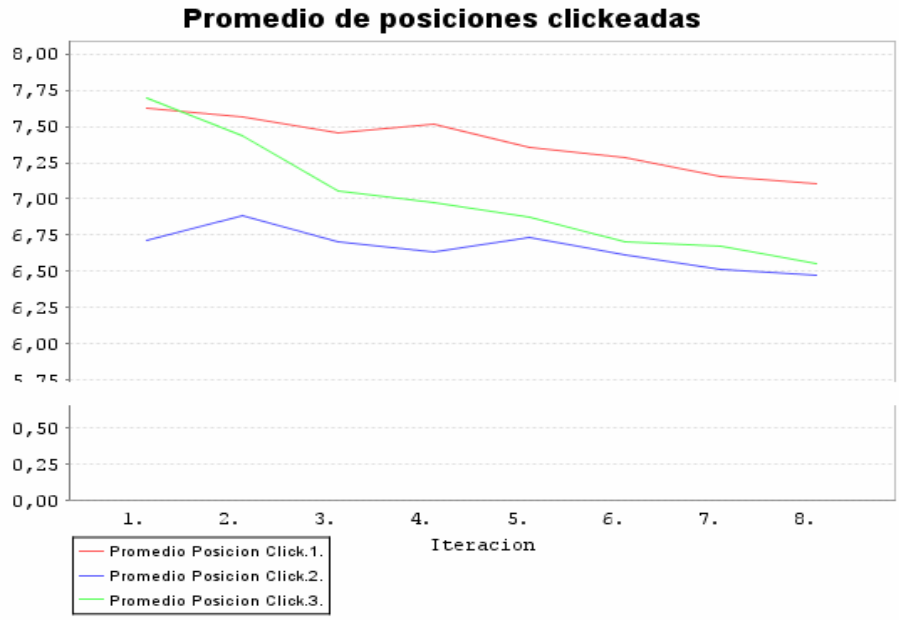

Figura 6: Promedio de posiciones clickeadas según algoritmos a través del tiempo

Asignar de distinta forma el peso de los documentos de los cuales no se tiene historial. Todos los documentos están interrelacionados, esto significa que, si un documento dentro de su contenido cita a otro documento, los mismos tienen un vinculo que los une, se puede imaginar como un grafo de documentos los cuales están todos interconectados, una mejora sería, que el peso del nuevo documento sea una media de los pesos de los documentos de los cuales él hace referencia.
Otro estudio puede estar enfocado a la clusterización de usuarios; el sitio contenía información jurídica, ahora se ha expandido, la idea es abrirse hacia a una plataforma global. Es decir, la información jurídica almacenada no solo del país estudiado sino de varios países. Esto implica un fuerte crecimiento de usuarios a nivel mundial. Las áreas dentro del sitio Web cambian considerablemente dependiendo el país, y para los perfiles de usuarios esto también varía. La idea apunta a agrupar usuarios por sectores del planeta con la opción de ir bajando niveles, por ejemplo, oriente-occidente, continentes, grupo de países, país etc. Los países podrían agruparse también en base a la similitud que tienen en un contexto legal, luego, fijado el nivel al que uno quiere llegar, clusterizar los usuarios de los grupos previamente armados.
Por ultimo, se podría calcular el peso de un documento, pero no solo por el CTR, sino por varios factores que engloban el comportamiento del usuario. Se puede considerar acciones como, si un documento se imprimió, si fue descargado, si fue enviado por mail a un colega o incluso si lo almaceno en alguna carpeta del propio sitio. Estas acciones generarían feedback implícito evitando tener que interrumpir al usuario para preguntarle si un documento es de interés o no para él.

REFERENCIAS

[1] Alfaro Arancibia R. Modelado de un Sitio Web para Mejorar la relación con los Usuarios. Escuela de Ingeniería Industrial de la Universidad Católica de Valparaíso. 2003. Disponible en http://www.rodrigoalfaro.cl/documentos/Articulo_Ingenerare.PDF

[2] Becker H., Meek C., Maxwell D. Chickering Modeling Contextual Factors of Click Rates. In Proceedings of Association for the Advancement of Artificial Intelligence, pages1310–1315. 2007.

[3] Cooley R., Srivastava J., Mobasher B. Web Mining: Information and Pattern Discovery on the World Wide Web. November 1997.

[4] Gospodnetic O. and Hatcher E. “Lucene In Action” Manning Publications Co. 2005

[5] Greengrass Ed. Information Retrieval: A Survey. November 2000. Disponible en: http://www.csee.umbc.edu/cadip/readings/IR.report.120600.book.pdf

[6] Ingersoll G. and Yilmazel O. Advanced Lucene. Presented at ApacheCon Europe 2007.

[7] Immorlica N., Jain K., Mahdian Mo., and Talwar K. Click Fraud Resistant Methods for Learning Click-Through Rates. LECTURE NOTES IN COMPUTER SCIENCE. NUMB 3828, pages 34-45. SPRINGER-VERLAG 2005

[8] Joachims T. Optimizing Search Engines using click through Data. *SIGKDD* 2002 Edmonton, Alberta, Canada.

[9] Joachims, T.; Granka, L. A.; Pan, B.; Hembrooke, H.; and Gay, G. Accurately interpreting clickthrough data as implicit feedback. In SIGIR 2005, pages 154–161.

[10] Min Zhao, Hang Li, Adwait Ratnaparkhi, Hsiao-Wuen Hon, Jue Tang Adapting Document Ranking to Users' Preferences using Click-through Data *CIKM'04*, November 8–13, 2004, Washington D.C., U.S.A. MSR-TR-2006-15 (TR-2006-15.doc)

[11] Radlinski F. and Joachims T. Minimally Invasive Randomization for Collecting Unbiased Preferences from Click through Logs. Proceedings of the 21st National Conference on Artificial Intelligence. 2006

[12] Raman K. The Laplace Rule of Succession Under A General Prior. 2000. Disponible en: http://interstat.statjournals.net/YEAR/2000/articles/0006001.pdf

[13] Villena J., González J. C., Barceló E., Velasco J. R. *Minería de uso de un servidor web mediante huellas y sesiones*. IBERAMIA 2002.

[14] Welcome to Lucene. Disponible en: http://lucene.apache.org/