TW – Engine: Motor de búsqueda de alto desempeño en Java

Weckesser, Carlos
Tomasino, Eugenio Enrique

Universidad Tecnológica Nacional, Facultad Regional Córdoba

Abstract

El buscador que se presenta en este trabajo es el resultado de un arduo proceso de investigación en el campo de los Motores de Búsqueda. Para el desarrollo del mismo se destaca el empleo de diversas técnicas de almacenamiento y recuperación de datos desde medios externos (ej: Disco Duro). Estas últimas a su vez se pueden implementar con un grado de performance notable gracias a la aplicación de Buffering.
Por otro lado, se emplean técnicas de ordenamiento ampliamente reconocidas y recomendadas (Quick-sort) para reducir en gran proporción los tiempos de procesamiento.
Adicionalmente ha de destacarse el desarrollo de un método propio para calificar y asignar relevancia a los resultados de las búsquedas, el cual ha devuelto resultados totalmente positivos.
El buscador se presenta tanto a usuarios como administradores por medio de una interfaz Web lo cual permite no solo su consulta sino también su mantenimiento de forma remota.
Además puede implementarse inmediatamente en cualquier organización que cuente con una base de datos descriptiva de los elementos (libros, revistas, productos, personas, artículos, servicios, etc) que serán, en última instancia, el objeto de las búsquedas.

Palabras Clave

- Acceso directo a medios externos (seeking): hace referencia al método que permite acceder a un medio externo (Ej. Disco duro) con total control de la posición y la forma en que se leen y escriben los datos.
- Archivos de posteo: archivos donde se guardan las listas de posteo. Estos archivos permiten al buscador guardar y tomar la información necesaria en tiempos muy reducidos y por lo tanto responder con gran velocidad a las consultas y a las actualizaciones. Esta velocidad se incrementa si los archivos de posteo se encuentra ordenados.
- Buffer: espacio en memoria que se emplea como transición entre distintos medios de almacenamiento. (Ej. memoria <> disco).
- Buffering: técnica del empleo de buffers. Específicamente, en la aplicación se emplean buffers en memoria que permiten reducir considerablemente la cantidad de accesos al disco duro. (Se escribe en el buffer una cierta cantidad de datos antes de realizar el acceso a disco)
- Celda de posteo: unidades que conforman las listas de posteo.
- Documento XML: documento que respeta el formato de etiquetas de XML (Lenguaje de Marcado Extendido).
- Índice de vocabulario: conjunto de todos los términos/vocablos que puede encontrarse en la base de datos empleada por el buscador.
- Lista de posteo: conjunto de elementos (celdas) que permiten identificar los objetos de búsqueda donde se encuentra un término o vocablo determinado.
- Objetos de búsqueda: elementos que serán al mismo tiempo los objetivos y los resultados de las búsquedas (Ej.: libros, revistas, productos, personas, artículos, servicios, etc.) de acuerdo a como haya sido implementado el buscador. En la presente versión, los objetos de búsqueda son libros.
- Recuperación de datos: ver introducción.
- Recuperación de información: ver introducción.
- Stopwords: términos que se encuentran en gran cantidad de objetos de búsqueda. Ej: suponiendo una base de datos de libros, sería de esperarse que gran cantidad de estos libros tengan en sus prólogos algún artículo, como ser “el”.
- Término: vocablo identificable dentro de algún campo de la base de datos (ej: nombre de persona, cualquiera de las palabras contenidas en el título de un libro, etc.)
- Vocabulario de términos: véase Índice de vocabulario.

Introducción

Los motores de búsqueda (o simplemente "buscadores") son una de las piezas fundamentales de la actual era de la información, dado que han facilitado el acceso a la información tanto en la web, como en bibliotecas, comercios y otras instituciones.
En lo que a búsquedas concierne, se han presentado dos casos:

a) Recuperación de datos
b) Recuperación de información

En el primer caso, tanto los datos almacenados (por lo general en una base de datos) como los lenguajes de consultas están organizados y estructurados. De esta forma el motor de búsqueda devuelve siempre al usuario, de manera exacta la información correspondiente a la consulta efectuada. (Ej. Búsqueda de una persona consultando una base de datos por medio del número de documento).
En el segundo caso, la información consiste en un conjunto de documentos de distinto tipo y sin ningún tipo de estructuración. En este contexto, las consultas (también desestructuradas) se intentan responder con los documentos que mejor se adapten a las mismas.

En el presente trabajo de investigación se ha abordado la problemática planteada en el segundo caso (recuperación de información) sin dejar de lado factores como la rapidez y la relevancia de los resultados a las búsquedas los cuales son, para cualquier buscador, determinantes en su éxito.
Para poder llegar a una solución que responda al problema de recuperación de información, técnicamente se requiere de dos pasos:

1) El diseño de un modelo matemático que describa formalmente la estrategia a emplear y le de sustento.
2) Diseñar, programar y testear los algoritmos y estructuras de datos necesarios para implementar el modelo.

Existen tres modelos clásicos, los cuales son:

a) Modelo Booleano
b) Modelo Vectorial
c) Modelo Probabilístico

a) El modelo Booleano fue el primero en plantearse. En este la relevancia de un documento ante una consulta es binaria ya que el documento es relevante si posee el término, o irrelevante si no lo posee. Puede observarse que este modelo presenta varias dificultades, como no permitir discriminar entre documentos más o menos relevantes, no considera el número de ocurrencias del término en el documento, entre otras.
b) El modelo vectorial consiste, muy básicamente, en considerar a cada documento como un vector de k componentes (uno por término). También considera a las consultas como vectores en sí mismas. De esta forma, se intenta determinar que vectores correspondientes a los documentos son más similares al vector consulta. Lo usual es medir el ángulo que forman dos vectores (cada uno de los vectores documento con el vector consulta), por medio del coseno de ese ángulo. Si el valor del coseno de ese ángulo es uno entonces el valor del ángulo es cero, con lo cual los vectores son muy similares. Si el valor del coseno es cero, los ángulos son perpendiculares lo cual significa que el documento y la consulta no comparten términos.
c) En el modelo Probabilístico se calcula el coeficiente de similitud entre una pregunta y un documento como la probabilidad que el documento sea pertinente a la pregunta. Esto reduce el problema de la pertinencia de un documento como una aplicación de la teoría de la probabilidad.

El desarrollo de este proyecto de investigación tiene como base al modelo vectorial de recuperación de información.
La aplicación de dicho modelo, se hizo en el marco de un motor de búsqueda que realiza consultas sobre un conjunto de documentos XML. Puntualmente, el programa de aplicación que se presenta consiste en un sistema de consultas de libros permitiendo optar por distintos criterios de búsqueda, ya sea buscando términos que pudiera contener un libro en sí, como también por nombre de autor, año de edición, título e ISBN.

Elementos del Trabajo y metodología

Se implementará el modelo Vectorial mediante el diseño de estructuras de datos y algoritmos apropiados. A los fines prácticos, no se recurrirá necesariamente a conceptos del álgebra vectorial tales como vector, espacio vectorial, etc., sino que se utilizarán estructuras que contienen a los datos y que de manera indirecta representan al modelo. De esta forma se lograrán realizar búsquedas rápidas sin tener que emplear elementos matemáticos complejos.
Concretamente, las estructuras de datos básicas implementadas son (véase Figura 1):

a) El vocabulario, que consiste en una tabla donde figuran todos los términos que forman parte de los documentos.
b) Listas de posteo, que por cada término del vocabulario agrupan los descriptores de los documentos que poseen ese término. Estos descriptores permiten identificar de manera unívoca cada archivo.

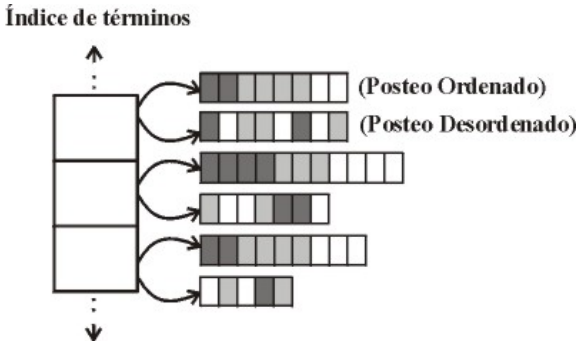

Figura 1

El vocabulario es colocado en la memoria RAM y allí permanece mientras esté activa la aplicación; sin embargo, las listas de posteo se almacenan en disco (el tamaño de estas listas pueden crecer indefinidamente, imposibilitando su almacenamiento en memoria).
Por más extenso que el vocabulario pueda parecer, siempre puede mantenerse en la memoria principal. Esta postura se defiende mediante una ley, denominada "Ley de Heaps" la cual se describe a continuación:

Si se analiza un documento de texto puede verse que el número de términos diferentes que aparecen en un principio es alto, pero que al ir avanzando en el documento, la cantidad de términos nuevos que irán apareciendo será cada vez menor. Si se observa la figura 2, se puede apreciar que el tamaño de un documento y la cantidad de términos (vocabulario) mantienen una relación logarítmica.

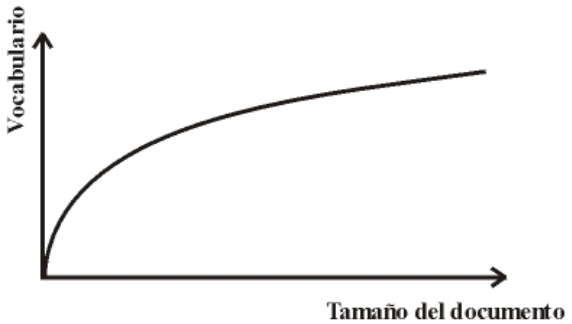

Figura 2

El vocabulario conforma un índice, donde por cada término puede encontrarse un conjunto de características que permitirán determinar, para ese término, cuales

documentos son relevantes en el momento de responder a una consulta. En este índice se han dispuesto para cada una de sus entradas los siguientes campos (véase figura 3):

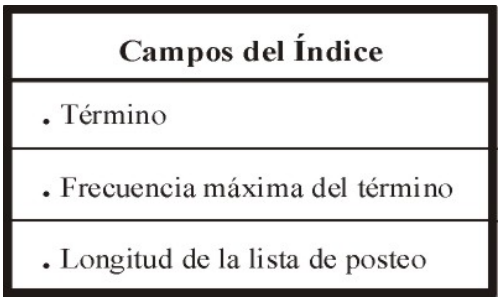

Figura 3

Se puede identificar al término en cuestión (Término), el mayor número de ocurrencias del término entre todos los documentos que lo contienen (Frecuencia máxima del término) y la cantidad de documentos que contienen a ese término (Longitud de la lista de posteo).

La lista de posteo de un término esta conformada por un grupo de celdas (una por cada documento que contenga a ese término), las cuales poseen dos elementos: una referencia al archivo de datos (el documento XML, en este caso) que permita identificarlo y la cantidad de veces (o frecuencia) que el termino aparece en dicho documento (véase figura 4).

Figura 4

En la figura 1 puede apreciarse que por cada término se cuenta con dos listas de posteo, una lista de posteo ordenada y otra de posteo desordenada.
Cuando la aplicación entra en funcionamiento por primera vez y se deben cargar los datos para permitir su funcionamiento (es decir, analizar los documentos confeccionando el vocabulario y las listas de posteo correspondientes que dan soporte al proceso de búsqueda), los documentos son procesados y las listas de posteo por cada término se colocan en el archivo de forma desordenada (se obtiene así una lista de posteo desordenada). Luego que han sido procesados todos los documentos seleccionados por el administrador de la aplicación, puede llevarse a cabo el proceso de ordenamiento, el cual dispone todas las celdas de la lista de posteo de forma adyacente (con lo cual se obtiene una lista de posteo ordenada).
Si se agregaran nuevos documentos a la "base de datos" de la aplicación, el posteo correspondiente a dichos documentos se generará en un archivo paralelo de manera desordenada. De esta forma para algunos términos habría tanto una lista de posteo ordenada como una lista de posteo desordenada (y que es lo que se aprecia en la figura 1).
Esta separación que se lleva a cabo se debe a que agregar nuevas celdas en una lista de posteo ordenada requiere de un mayor tiempo de procesamiento para redimensionar y reacomodar el archivo de datos. El hecho de agregar nuevos documentos, colocando las celdas correspondientes en un nuevo archivo hace posible que la base de datos puede seguir creciendo sin ocasionar demoras innecesarias en el servicio del buscador. Si bien contar con un conjunto de datos desordenados produce una disminución del desempeño de la aplicación en el proceso de búsqueda, esta es mínima ya que no se tardaría tanto tiempo en procesar esa pequeña cantidad de datos desordenados. Si además se añade el hecho de que la base de datos puede ordenarse en un momento en que la aplicación tenga un uso mínimo o nulo, esto hace que el desempeño solo se vea afectado períodos de tiempo limitados.

El motor inicia su procedimiento de búsqueda cuando se efectúa una consulta, tomando cada uno de los términos de dicha consulta por separado. Por cada uno de

ellos se considera tanto a ese término, como a todas las posibles formas en que pudieran aparecer en un documento bien redactado (es decir, sin errores ortográficos). Estas variaciones consisten en considerar al término que aparezca rodeado de cualquier signo habitual como los de puntuación, exclamación, interrogación, entre otros. A continuación, se procede a efectuar la búsqueda de cada uno de los términos (tanto los de la consulta como sus variaciones) en el vocabulario.
Una vez obtenido el conjunto de todos los documentos que contienen al menos uno de los términos de la consulta, se debe proceder a establecer una ponderación y una clasificación descendente, en la cual los documentos más relevantes para esa consulta se encuentren en las primeras posiciones. Para ello se contemplan los siguientes parámetros:

n: Cantidad total de documentos que conforman la base de datos
lp_i: Longitud del posteo del término i (cantidad de documentos que contienen al término i)
f_{ik}: Frecuencia del término i en el archivo k
L_k: Longitud o tamaño del archivo k (en bytes)
C_k: Cantidad de términos de la consulta que aparecen en el archivo k
s: Cantidad de términos en la consulta

A partir de estos, es posible calcular un valor que indica la importancia de cada documento.
El cálculo de este índice se propone por medio de la siguiente fórmula:

$$I = \sum_{i=1}^{s} [(1 - lp_i/n) * f_{ik}] * C_k/L_k$$

En esta potente fórmula se consideran varios aspectos cruciales para establecer la relevancia de un documento. El cociente entre la longitud de posteo de un término y la cantidad total de documentos permite detectar si el término es un stopword o no, dependiendo si el resultado es cercano a cero o cercano a uno respectivamente. Al multiplicar por la frecuencia de un término en un documento, documentos que posean un mayor número de veces al término tendrán un mayor puntaje. Si luego al resultado parcial (ya efectuada la sumatoria que contempla a todos los términos de la consulta que se encuentran en el documento) se multiplica por la cantidad de términos de la consulta que figuran en el documento, aquellos que posean más términos de la consulta, se consideran más relevantes. Por último, al dividirse por el tamaño del documento se logra que aquellos documentos que posean igual puntaje parcial pero que tengan menor tamaño sean aún más relevantes, ya que se considera que la densidad de información relativa a la consulta es en estos mucho mayor.

Resultados

El resultado general del trabajo realizado se puede enunciar como el de haber implementado un motor de búsqueda capaz de generar, a partir de un conjunto de archivos XML convenientemente estructurados, su propio vocabulario de términos y sus propias listas de posteo quedando listo para ponerse en funcionamiento.
Durante el proceso de desarrollo del buscador han aparecido numerosos resultados parciales, los cuales se corresponden con cada una de las ideas y propuestas que se proponía implementar. Dichos resultados se listan a continuación:

- Presentar el buscador por medio de una interfaz Web, tanto a usuarios como a administradores: esta interfaz Web permite consultar y administrar el buscador de manera remota desde cualquier ordenador que tenga acceso al servidor donde se encuentre instalada la aplicación.

- Implementar una base de datos en forma de un conjunto de archivos XML: de esta forma se logra independizar al buscador de cualquier motor de base de datos relacional. La información que necesita el usuario se guarda en archivos XML convenientemente ubicados en un directorio que a su vez es controlado por el buscador. Este formato a su vez va de la mano con la implementación web de la interfaz permitiendo mostrar al usuario directamente los documentos consultados (sin necesidad de realizar una nueva consulta, lo cual sería necesario si la información se encontrara, por ejemplo, en una base de datos relacional).
- Generar y utilizar archivos de posteo ordenados y sin apropiación innecesaria de espacio de disco: para disponer de listas de posteo ordenadas en un medio externo de almacenamiento se presentan dos opciones bien diferenciadas; por un lado (ver Figura 5.b) se puede reservar para cada término del buscador un espacio de disco tal que pueda albergar la totalidad de la lista de posteo (habiendo previsto el crecimiento de dicha lista a lo largo de la vida del buscador). Por otro lado (ver Figura 5.a), y siendo esta la opción implementada finalmente, se puede pensar en ubicar las listas de posteo ordenadas de forma contigua. Si bien esto ahorra notablemente el espacio de disco utilizado, surgen complicaciones en el momento de agregar nuevos objetos de búsqueda (en este caso, archivos XML con descripciones de libros) a la base de datos ya que esto implicaría generar nuevamente el archivo de posteo. El buscador esta diseñado para permitir a su administrador cargar información en un determinado momento y ordenarla en otro posterior; así, contemplando como ejemplo la situación en la que el buscador ya maneja una gran cantidad de información y se quiere agregar una nueva porción de datos, este anexado se puede realizar de forma inmediata, dejando el proceso de ordenado de los posteos (que consumirá mas tiempo y que requerirá deshabilitar el servicio del buscador durante un período de tiempo mas extenso) para otro momento.

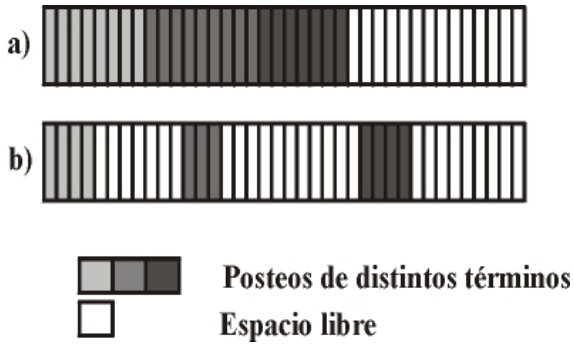

Figura 5

- Mejorar considerablemente las demoras ocasionadas por los accesos a disco: el motor de búsqueda es una aplicación que necesita manejar una gran cantidad de información, y es por esto que resulta impensable disponer de todos los datos necesarios en la memoria. Por otro lado, se sabe que los accesos a disco (actualmente el medio de almacenamiento externo mas popular) llevan una importante cantidad de tiempo. Empleando una técnica de buffering se ha conseguido reducir a un mínimo aceptable la cantidad de accesos a disco. Esta mejora se pudo apreciar, por ejemplo, en la notable reducción de los tiempos de generación de los archivos de posteo.

Efectuar el proceso de ponderación de los resultados de las búsquedas en tiempos muy

reducidos: luego del proceso de búsqueda propiamente dicho, el buscador se encarga de seleccionar aquellos resultados que considera, serán de mayor interés para la persona que solicitó la búsqueda. Esta clasificación se lleva a cabo gracias a la implementación de un modelo matemático propio. (Este último se describe con mayor detalle al final del inciso Elementos del Trabajo y metodología)

Discusión

Muchos de los resultados expuestos en el inciso anterior se relacionan y/o complementan mutuamente. Estos vínculos se describen a continuación:

La técnica de buffering permite a la aplicación responder con mayor velocidad en los principales procesos que esta ejecuta, que son: el proceso de búsqueda, el proceso de generación de las listas de posteo y el proceso de ordenamiento de dichas listas.
El proceso de búsqueda también se vio mejorado por el empleo del modelo matemático que le permite ordenar velozmente los resultados obtenidos.
La creación de archivos XML para contener la información manejada por el buscador puede adaptarse con mayor facilidad a la interfaz web con la que se presenta el buscador. A la hora de mostrar los resultados, el buscador solo se limita a mostrar, por medio del navegador web, el archivo XML que contiene la información solicitada.

Por otro lado, si bien la presente aplicación se muestra como un buscador que podría brindar sus servicios a una biblioteca o a una librería, la base de la aplicación (el motor de búsqueda propiamente dicho) se podría adaptar y generalizar a cualquier situación en la que se tenga la necesidad de búsqueda y recuperación de información. Otro aspecto importante de la aplicación que resulta posible de ser generalizado es la técnica de buffering. Esta última podría emplearse y a la vez mejorar cualquier aplicación que haga uso exhaustivo del acceso directo a medios externos como discos duros, diskettes, etc.

Un tema importante en materia de motores de búsqueda es el de las “stopwords” (Véase palabras clave). El tratamiento de stopwords empleado por el buscador consiste en lo siguiente: cuando un usuario realiza una consulta introduciendo una stopword, el buscador considera solo un número reducido de los documentos donde ésta aparece. Luego, en el proceso donde los resultados son clasificados y ponderados, estos documentos que contienen dicha stopword con mayor frecuencia serán desplazados por los documentos que contengan las palabras significativas de la búsqueda (en el caso de que estas existan en la consulta). Como puede deducirse, el buscador dará poca importancia a los documentos que contengan únicamente stopwords, pero no los dejará totalmente de lado (abriendo la posibilidad a consultas con stopwords predominantes).

Por último, se pueden realizar diversas comparaciones que permiten apreciar la tecnología empleada en el presente trabajo. A continuación se procede a describir algunas comparaciones que ponen en evidencia las ventajas y desventajas de haber seleccionado las técnicas empleadas:

- El empleo de buffering ha incrementado notablemente la velocidad general de la aplicación. Esto se ha podido comparar directamente con versiones anteriores de la aplicación donde todavía no se había implementado buffering. Se ha logrado, por ejemplo, disminuir el proceso de creación de los archivos de posteo de varias horas a unos pocos minutos.
- El hecho de ordenar las listas de posteo de forma contigua en un archivo tiene la desventaja de que

requiere una mayor cantidad de tiempo para insertar nuevos datos. (Se deben levantar todos los posteos, ordenarlos y volverlos a bajar; este problema se ha tratado como se describe en los resultados). Sin embargo permite un aprovechamiento muy eficiente del espacio de disco (lo cual no ocurriría si se reservara espacio para albergar los crecientes posteos de cada término).

- La potencia del buscador permite, trabajar tanto en archivos de posteo ordenados como en archivos de posteo desordenados, sin que sea necesario realizar el ordenamiento de dichos posteos al momento de agregar información a la base de datos, sino que los nuevos datos se introducen ágilmente de manera desordenada y el buscador, aunque con una leve demora, podrá continuar su normal funcionamiento respondiendo a consultas, e incluso devolviendo en sus resultados la información recientemente ingresada. El administrador podrá elegir el mejor momento para llevar a cabo el ordenamiento que restaurará el normal funcionamiento del buscador. Si se obligara al buscador a ordenar en cada instancia de actualización, se debería escoger entre cortar el servicio del buscador en cualquier momento que se quieran agregar datos o bien dejar las actualizaciones para un momento en el que el servicio del buscador no sea tan requerido.
- El tratamiento de stopword empleado brinda la posibilidad de contemplar consultas formadas por stopwords o donde estas predominan por sobre el resto de los términos. En otros buscadores, donde las stopwords son directamente descartadas de la búsqueda, una consulta como la anteriormente mencionada no devolverá ningún resultado. Otra opción sería dejar de lado el tratamiento de las stopwords, lo cual sobrecargaría notablemente el procesamiento innecesario de datos dentro del servidor.

Conclusión

El presente trabajo consiste en un motor de búsqueda con interfaz web adaptable a distintas áreas de aplicación. Con motivo de su exposición al público, se encuentra implementado como un buscador para biblioteca/librería, permitiendo al usuario efectuar consultas de libros en base a distintos criterios (contenido, autor, ISBN, titulo, etc).
Son de destacarse aspectos como la velocidad de búsqueda, la potencialidad de administración y el aprovechamiento de espacio en disco. Dichos aspectos destacan esta aplicación posicionándola en un importante nivel de competencia.
La primera versión de este trabajo ha sido propuesta como entrega final de la materia Diseño de Lenguajes de Consulta. Posteriormente se han reformulado radicalmente numerosas secciones de la aplicación permitiendo incrementar aún mas la potencia y la performance de la misma.

Agradecimientos

Se agradece principalmente, al Ing. Valerio Frittelli por la constante dedicación que ha demostrado a lo largo de todas las etapas de formación en las cuales hemos tenido el privilegio de contar con él como docente. Por la calidez del ambiente que ha logrado mantener a lo largo de sus clases y sobre todo por la motivación que nunca ha dejado de transmitir.

Datos de Contacto:

Eugenio Enrique Tomasino. Universidad Tecnológica Nacional, Facultad Regional Córdoba. Bahía Blanca 529 Bº Juniors CP 5000. *etomasino@gmail.com*

Carlos Weckesser. Universidad Tecnológica Nacional, Facultad Regional Córdoba. Avenida Colón 2656 Bº Alto Alberdi CP 5003.
carlosweckesser@hotmail.com