A New Information Filtering Method for WebPages

(joint work with Sergio López)
Contents

Motivation
• Information Filtering
• Information Retrieval
Filtering Single Webpages
• Basic Algorithm
Filtering Multiple Webpages
• The Hyper-Syntactic Distance
Reconstructing Retrieved Information
• Tabular vs Hierarchical Views
Conclusions and Future Work
Motivation

Retrieving information from the Web (a single webpage)

Any webpage
It can include frames or iframes
No preprocessing stage
No semantic information
No microformats
No ontologies
etc.
Motivation

Retrieving information from the Web (a single webpage)

How do you extract non-textual information (images, videos, etc.)? How do you keep the original structure of the webpage?
Motivation

What is the Web Filtering Toolbar?

It is an official Firefox’s plugin whose interface is a toolbar

http://www.firefox.com/addons

Start the filtering process

Filtering criterion
How much information do you want?
Show everything
Motivation

Retrieving information from the Web (multiple webpages)

What pages do you analyze?
Are domains important?
What about security?
What about layers?
Motivation

Objectives

1. The technique should be able to work online
   - No precompilation phases
   - No proxies

2. The technique should be able to work with any webpage
   - Unknown objects must be processed
   - No restrictions on the language
   - No semantic information is provided to the technique (just syntactic)
     - Forget about ontologies
     - Forget microformats
Contents

Motivation
• Information Filtering
• Information Retrieval

Filtering Single Webpages
• Basic Algorithm

Filtering Multiple Webpages
• The Hyper-Syntactic Distance

Reconstructing Retrieved Information
• Tabular vs Hierarchical Views

Conclusions and Future Work
Filtering Single Webpages

The Technique

A web page can be seen as a tree of labeled nodes.

Internally represented in DOM with a data structure called “document”

We use this model to see web pages as a hierarchically organized collection of nodes.
Filtering Single Webpages

The Technique
Filtering Single Webpages

The Technique

We first construct a tree-like data structure which contains all the information of the web page hierarchically organized in document order.

With the API of DOM we can explore the nodes of a web page and query their properties.
Filtering Single Webpages

The Filtering Criterion

Text

(words; flags)

Vector with flags for the filtering tool

Options Web Slicing Toolbar

- Keep Structure
- Format Size iFrames
- Keep Tree

Tolerance: 0

Acceptar | Cancelar
Filtering Single Webpages

The Filtering Criterion

Example: \((\text{cars}; [1,1,1,0])\)

1) We compare “cars” with the information contained in the attributes of each node.

2) For special nodes (such as images) we use different attributes (e.g., “alt”)

3) Nodes containing “cars” are called relevant.

4) We proceed by marking those nodes which are related to relevant nodes.

5) Not marked nodes are deleted or hidden.
Example: Deleting nodes

1. we could delete all the nodes except 5 (Firefox’s search)
2. or all the nodes except 5, its descendants and its ancestors
3. or except 5 and its descendants
Filtering Single Webpages

The Filtering Technique

Diagram 1

Diagram 2
The Filtering Technique

To summarize, we filter a given web page with respect to a filtering criterion by

(i) Building a data structure containing all the elements in the web page,

(ii) Traversing the data structure to find the relevant nodes with respect to the filtering criterion, and

(iii) Deleting (or hiding) all the nodes from the web page except the relevant nodes according to the filtering criterion's flags.
Contents

Motivation
  • Information Filtering
  • Information Retrieval
Filtering Single Webpages
  • Basic Algorithm
Filtering Multiple Webpages
  • The Hyper-Syntactic Distance
Reconstructing Retrieved Information
  • Tabular vs Hierarchical Views
Conclusions and Future Work
The Hyper-Syntactic Distance

Definition 5 (Search Hyperspace). Given a webpage P=(u,t) the search hyperspace of P is the set of webpages that either are reachable following hyperlinks from nodes of P, or that can reach P from their hyperlinks.

The search hyperspace is the collection of webpages that are related to the initial webpage, and that should (ideally) be inspected by our information retrieval algorithm. However, the search hyperspace of a webpage is potentially infinite (even more when we surf dynamic webpages [11]), and it is often huge.
Filtering Multiple Webpages

The Hyper-Syntactic Distance

What links should we inspect and in what order?
Filtering Multiple Webpages

The Hyper-Syntactic Distance

- **DOM distance (d\text{T})**: It is the length of the path between two nodes of a DOM tree.

- **Page distance (d\text{P})**: It is the lower number of hyperlinks that must be traversed to reach one webpage from another webpage.

- **Domain distance (d\text{D})**: It is the lower number of domains that must be traversed to reach one webpage from another webpage following a path of hyperlinks.
Filtering Multiple Webpages

The Hyper-Syntactic Distance

**Definition 6 (Hyper-Syntactic Distance, Relevance).**
Given a DOM node \( n \), the hyper-syntactic distance of \( n \) is:

\[
D = dT + KP \cdot dP + KD \cdot dD
\]

where \( KP \) and \( KD \) are numeric constants used to weight the equation.

The relevance \( R \) of a DOM node is the inverse of its hyper-syntactic distance:

\[
R = \frac{1}{D}
\]

The constants \( KP \) and \( KD \) determine the importance that we give to the fact that the word specified by the user is in another page, or in another domain.
Filtering Multiple Webpages

The Hyper-Syntactic Distance

\[ D = d_T + 10^6 \times d_P + 10^9 \times d_D \]

\[ 3 + 0 \times 10^6 + 0 \times 10^9 = 3 \]
\[ 1 + 0 \times 10^6 + 0 \times 10^9 = 1 \]
\[ 1 + 1 \times 10^6 + 0 \times 10^9 = 1 + 10^6 \]

\[ \frac{1}{3} = 0.333 \]
\[ \frac{1}{1} = 1 \]
\[ \frac{1}{10^6} = 0.000001 \]
Filtering Multiple Webpages

The Hyper-Syntactic Distance

<table>
<thead>
<tr>
<th>Hyperlink</th>
<th>dT</th>
<th>dP</th>
<th>dD</th>
<th>D</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1000000</td>
<td>0.000001</td>
</tr>
</tbody>
</table>
Contents

Motivation
  • Information Filtering
  • Information Retrieval
 Filtering Single Webpages
  • Basic Algorithm
 Filtering Multiple Webpages
  • The Hyper-Syntactic Distance
 Reconstructing Retrieved Information
  • Tabular vs Hierarchical Views
 Conclusions and Future Work
Visualizing Algorithms

Reconstructing Retrieved Information
Visualization Algorithms

Algorithm 1 Tabular Visualization

Input: A webpage $P$, and a filtering criterion $q$
Output: A webpage $P'$
Initialization: $link = \emptyset$

function show(Node $n$, DOM $d$)
    showTabular($n$, $d$)

function showTabular(Node $n$, DOM $d$)
    $iframe = createFrame(d)$
    append($n$, $iframe$)

function processWebPage(Link $l$, WebPage $p$)
    relevantNodes = getSlice($p$, $q$)
    if ($l \neq \emptyset$)
        then nodeC = getNode($l$)
    else nodeC = $<BODY>$
    show(nodeC, relevantNodes)
    links = links $\cup$ getLinks(relevantNodes)
    if ($timeout() \lor links = \emptyset$)
        then exit()
    else link = getMostRelevantLink(links, $q$)
        links = links \ link
        newPage = load(URL2) where link = (URL1, URL2)
        processWebPage(link, newPage)

return
    $<HTML>$
    $<BODY>$
    processWebPage(link, $P$)
    $<HTML>$
    $<BODY>$
Reconstructing Retrieved Information

Visualization Algorithms

Algorithm 2 Hierarchical Visualization

Input: A webpage \( P \), and a filtering criterion \( q \)
Output: A webpage \( P' \)
Initialization: \( link = \emptyset \)

function show(Node \( n \), DOM \( d \))
    showHierarchical(\( n \), \( d \))

function showHierarchical(Node \( n \), DOM \( d \))
    container = createContainer()
    if \( (n \neq < BODY >) \)
        then append(getParent(\( n \)), container)
    else append(\( n \), container)
    append(container, \( d \))

return
    \(< HTML >\)
    \(< BODY >\)
    processWebPage(link, \( P \))
    \(< \backslash HTML >\)
    \(< \backslash BODY >\)
Reconstructing Retrieved Information

Visualization Problems

Layers

+ Layers use absolute positions: Overlapping is possible and frequent!!!
Reconstructing Retrieved Information

Visualization Problems

Security

+ Cross Site Scripting (XSS): An attacker could execute scripts from a page or domain different from the loaded webpage.

➤ Web browsers’ anti-XSS: DOM scripts of external pages are blocked.

✔ Use an iframe object:

```javascript
frame = document.createElement("iframe");
...
frame.setAttribute("type","content");
...
frame.webNavigation.allowJavascript = false;
frame.webNavigation.allowMetaRedirects = true;
frame.webNavigation.allowPlugins = false;
```
Design Decisions

+ The algorithm should never analyze a webpage with a page distance greater than 5. This is also supported by previous studies (see, e.g., Baeza and Castillo's discussion in [11]) that demonstrate that, in general, three to five levels of navigation are enough to get 90% of the information which is contextually related with the webpage selected by the user for the web search.

+ Hyperlinks that do not belong to the slice are discarded.

+ The maximum time spent to retrieve and show the information is 10 seconds. Usability rules [13] establish that 10 seconds is (as an average) the time limit that users spend waiting for a webpage to be loaded.
Performance Evaluation

<table>
<thead>
<tr>
<th>URL</th>
<th>Filtering criterion</th>
<th>Links visited</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.iee.org">www.iee.org</a></td>
<td>student</td>
<td>12</td>
</tr>
<tr>
<td><a href="http://www.upv.es">www.upv.es</a></td>
<td>student</td>
<td>18</td>
</tr>
<tr>
<td><a href="http://www.who.int">www.who.int</a></td>
<td>OMS</td>
<td>25</td>
</tr>
<tr>
<td><a href="http://www.un.org">www.un.org</a></td>
<td>Haiti</td>
<td>6</td>
</tr>
<tr>
<td><a href="http://www.esa.int">www.esa.int</a></td>
<td>launch</td>
<td>13</td>
</tr>
<tr>
<td><a href="http://www.nasa.org">www.nasa.org</a></td>
<td>space</td>
<td>16</td>
</tr>
<tr>
<td><a href="http://www.mec.es">www.mec.es</a></td>
<td>beca</td>
<td>18</td>
</tr>
<tr>
<td><a href="http://www.edu.gva.es">www.edu.gva.es</a></td>
<td>universitat</td>
<td>20</td>
</tr>
<tr>
<td><a href="http://www.ilo.org">www.ilo.org</a></td>
<td>projects</td>
<td>8</td>
</tr>
<tr>
<td><a href="http://www.unicef.es">www.unicef.es</a></td>
<td>Haiti</td>
<td>10</td>
</tr>
<tr>
<td><a href="http://www.mityc.es">www.mityc.es</a></td>
<td>turismo</td>
<td>9</td>
</tr>
<tr>
<td><a href="http://www.mozilla.org">www.mozilla.org</a></td>
<td>firefox</td>
<td>7</td>
</tr>
</tbody>
</table>

The average result is 13.5 webpages analyzed for each URL.

Note that a timeout of 10 seconds is the maximum time used to complete the final results webpage. But the visualization algorithms are incremental, thus, as an average, the first result is shown in less than a second (10/13.5 seconds).
Implementation

Distribution and installation

The Web Filtering Toolbar is distributed as an xpi file. An xpi package is basically a ZIP file that, when opened by the browser utility, installs a browser extension. Currently, this extension applies to both Mozilla and Firefox browsers.

Since an xpi package contains all the necessary information to install an extension, the user only has to drag the xpi and drop it over the browser window. The extension will be automatically installed.

To download the tool and some other materials visit:

http://www.dsic.upv.es/~jsilva/webfiltering

and

http://www.firefox.com/addons
Implementation

The Filtering Technique

The user interface of the *Web Filtering Toolbar as the Firefox's interface* itself has been implemented with *XUL* and *Javascript*.

*XUL* is an XML implementation which provides the interface components (such as buttons, menus, etc.);

*Javascript* is used to control and execute the user actions.

Both parts are independent and can be updated without affecting the other part of the tool.
Contents

Motivation
  • Information Filtering
  • Information Retrieval

Filtering Single Webpages
  • Basic Algorithm

Filtering Multiple Webpages
  • The Hyper-Syntactic Distance

Reconstructing Retrieved Information
  • Tabular vs Hierarchical Views

Conclusions and Future Work
Conclusions and future work

Conclusions:

• Current search engines consider web pages as the basic unit of information which should be provided to the user.

• From our point of view, this is frequently an error which makes the user to load, read and scroll a lot of useless information.

• We have introduced a new approach to web filtering:
  1. Automatically filter irrelevant information
  2. Reorganizing relevant information

• Implementation integrated into the Firefox browser
Conclusions and future work

Future Work:

• **Adapting the toolbar to other browsers:** The use of a standard DIV with standard Javascript can work in any browser. It is currently being adapted to Google Chrome.

• **Using a lexicon:** This will allow to filter web pages by using alternative filtering criteria semantically related to the one specified by the user.

  “cars” ≈ “auto” ≈ “automobile”

• **Filtering XML documents.** Some preliminary research in this direction has shown that our technique is directly applicable to XML documents. We would like to take advantage of the fact that with XML documents we have available a DTD which provides helpful information about the structure of the document.