A Method of Browsing Hierarchically Displayed Data in a Constrained Space

Francisco Corella

June 2010

1. Motivation

Computer systems often display data hierarchically and provide a user interface for browsing such hierarchically displayed data. Most operating systems, for example, have a hierarchical file system comprised of folders and files and provide a user interface for browsing the hierarchy of folders and files.

In the last few years, some Web applications have resorted to assigning tags to data such as bookmarks, photos, or email messages, as a means of organizing and retrieving such data [1]. Tagged data is not inherently hierarchical, but can be displayed and browsed hierarchically, as explained below in Section 3.

The social bookmarking application Delicious, for example, lets users assign tags to bookmarks, and displays tags and bookmarks hierarchically in a browser sidebar. Figure 1 shows a screenshot of a Mac computer where of a window of the Firefox browser occupies most of the screen. The Delicious sidebar, provided by the Delicious plug-in for Firefox, can be seen on the left side of the window.

The latest version of Noflail Search lets users run a given query on many different search engines, uses tags to organize the collection of engines, and displays tags and engines hierarchically in a panel entitled Search Engines, called the Engines panel. Figure 2 shows a screenshot of a Mac computer with Noflail Search running in a window of the Safari browser. The Engines panel is the second vertical panel from the left. Figure 3 show a close up view of the same Engines panel.

Sometimes a user interface must display a hierarchy of data in a constrained space. The Delicious Sidebar, for example, must be narrow because most of the space available in the browser window containing the sidebar must be dedicated to displaying Web pages. The size of the Engines panel of Noflail Search is constrained by the fact that it must share the space available to the Noflail Search application with up to three other panels and a number of other user-interface elements.

Traditional user interfaces methods of browsing hierarchical data are not well suited for use in a constrained space. Their drawbacks are most apparent when used to display hierarchies with a large branching factor. The present method of browsing hierarchical data that is well suited use in a constrained space, even when the branching factor of the hierarchy is very large.

Section 2 defines a few concepts pertaining to hierarchically defined data. Section 3 explains how tagged data can be browsed hierarchically, and points out that the branching factor of hierarchically browsed tagged data can be very large. Section 4

describes traditional methods of browsing hierarchical data and points out their drawbacks. Section 5 describes the new method of browsing hierarchical data.

2. Hierarchies

For the present purposes, a *hierarchy* can be defined as a (finite) collection of nodes, each node being either a data node (D-node) or a hierarchy node (H-node), and a parent-child relation among nodes such that a D-node has no children, and there are no directed cycles (i.e. there are no cycles where each node is a child of the previous node). A node can have zero, one, or more than one parents. A node with no parents is said to be a root. A node can also have zero, one, or more than one children. The number of children is called the branching factor of the node. The children of a node that are D-nodes will be called D-children, and those that are H-nodes will be called H-children.

The collection of folders (or directories) and files in a file system is an example of such a hierarchy, with the folders being H-nodes and the files being D-nodes. Notice that, while D-nodes have children, there can also be H-nodes without children (empty folders in the example).

3. Browsing tagged data as a hierarchy

Tags assigned to a data collection can be used to search the collection. For example, a user could enter a tag in a search box to retrieve all data items that have been assigned that tag. A user could also enter a set of tags to retrieve data items each of which has been assigned all of those tags.

But tags can also be used to build a hierarchy on the data, which will be called below the *tag hierarchy*, and allow the user to browse that hierarchy. The *tag hierarchy* can be defined as follows:

- The D-nodes are the data items.
- The H-nodes are the sets of co-occurring tags, the tags in a set being said to be co-occurring if and only if there exists a data-item tagged by all of them. The empty set is an H-node, which will be called the empty H-node.
- An H-node is a parent of a data item (a D-node) if and only if every tag in the H-node has been assigned to the data item. The empty H-node is a parent of those data items that have been assigned no tags.
- A first H-node is a parent of a second H-node if and only if the first H-node is not the empty H-node and the second H-node consists of all the tags of the first H-node plus one additional tag.

The roots of this hierarchy are the empty set, whose children are the data items with no tags, and the H-nodes consisting of only one tag. (The above definition explicitly excludes the empty H-node from being a parent of another H-node, as a matter of browsing convenience. Without that exclusion, the empty H-node would be the only root of the hierarchy.)

Tag hierarchies usually have large branching factors. Consider for example an ordinary file system with folders and files. A (larger) tag hierarchy can be derived from the folder-and-file hierarchy by assigning each folder as a tag to all the files

contained in the folder and its (recursively defined) subfolders. If the original folder-and-file hierarchy has a single root (the root folder), then the derived tag hierarchy has one root with a very large branching factor. The root of the derived tag hierarchy is an H-node consisting of a single tag, that tag being the root folder. All files are D-children of the root in the tag hierarchy; and, for every folder other than the root folder, the root of the tag hierarchy has an H-child consisting of that folder plus the root folder.

Hierarchies with large branching factors are difficult for users to browse. A goal of this invention is to facilitate the browsing of tag hierarchies and other hierarchies with large branching factors even within a constrained space.

4. Prior art methods of browsing hierarchical data

Several methods have been used in the past for browsing hierarchical data.

A first method consists of displaying together the H-children and the D-children of a current node in a single area of the display, and providing means of navigation to display the children of a different node in the same area. When the hierarchy is a file system, those means of navigation often include clicking or double-clicking on an H-node (a folder) to display the children of that H-node instead of those of the current node.

A second method is like the first method, except that clicking or double-clicking on an H-node displays the children of the H-node in a different area of the screen, multiple such areas being visible simultaneously.

A third method consists of displaying the hierarchy vertically as a list of nodes, and expanding or contracting the list as the user browses. An icon is associated with each node and can be in two states “closed” or “open”. When the user clicks on a closed icon, the icon changes to the open state, and the children of the H-node are inserted into the list, below the H-node and indented to the right. When the user later clicks on the item in the open state, the icon changes to the closed state and the children, or more generally the descendants, of the H-node listed after the H-node are removed.

A fourth method consists of displaying the hierarchy vertically as a list consisting only of H-nodes. Two icons are associated with each H-node. Clicking on one of the icons inserts into the list of nodes the H-children of the H-node. Clicking on the other icon has the same effect, plus the additional effect of displaying both the H-children and the D-children of the H-node in a separate area of the display.

A fifth method, used by the Delicious sidebar displays a vertical list of H-nodes, like the fourth method, one H-node to a line. Two icons may be associated with each H-node, but only one of them is active, the other being decorative. The inactive icon is an image of a kind of paper tag. The active icon is a triangle, which can be in a “closed” state pointing to the right or an “open” state pointing down. Inactive icons and active icons in the “open” and “closed” state can be seen in Figure 1. Clicking on an active icon in the closed state causes the icon to change to the open state and the H-children of the associated H-node to be inserted below the H-node, indented to the right. However these H-children do not have an associated active icon, and cannot be

further expanded to show their own H-children. Thus a non-indented line corresponds to an H-node that consists of a single tag, which is used to label the line; and an indented line corresponds an H-node that consists of two tags, the tag of the H-node in the closest non-indented line above the indented line, and an additional tag, which is used to label the indented line. Clicking anywhere on an H-node line, except on the active icon if one is present in the line, causes the line to become highlighted by a blue background and the D-children of the H-node to be displayed in a separate display area that occupies the bottom half of the sidebar. This area is labeled "Bookmarks", the D-nodes of the hierarchy being bookmarks of Web pages. Figure 1 shows the Bookmarks area containing the two D-children of the H-node consisting of the single tag "ItalianEnglish".

All these methods have substantial drawbacks when used in a constrained space, particularly when used for hierarchies with large branching factors.

The first method, because it relies on navigation, imposes on the user the burden of remembering the structure of the hierarchy to avoid getting lost. Furthermore, because H-children and D-children are displayed in the same area of the screen, if space constraints require that area to be small, H-children may be hard to find when there are too many D-children. For example, the user may not realize that H-children are present if none of them is visible without scrolling. Conversely, D-children may be hard to find if there are too many H-children.

The second method has the drawback of requiring multiple areas of the screen to be visible simultaneously, which may be hard to achieve in a constrained space, in addition to the second drawback of the first method.

The third method does not use navigation, and hence does not have the first of the drawbacks of the first method. But it has the second drawback of the first method: when the user clicks on an H-node, if the size available to display the vertical list of nodes is small due to space constraints, and the branching factor for the H-node is large, most of the children will not be visible without scrolling. D-children may then be hard to find if there are too many H-children, and H-children may be hard to find if there are too many D-children.

The fourth method requires two different areas of the display, which is a drawback if there are space constraints. Further it has the same drawback of displaying D-children and H-children together as the first and third methods.

The fifth method requires two different areas of the display. This is a drawback if space is scarce, as is the case for the Delicious sidebar. Because two different areas are required, only half of the sidebar is used to display the list of H-nodes.

5. Description of the present method

The present method can be used in a constrained space to browse hierarchies with large branching factors without the drawbacks of the prior art methods. In Noflail Search it is used to browse a tagged collection of search engines in the Engines panel as shown in Figure 3.

The Engines panel displays a tag-hierarchy, where the D-nodes are search engines such as “Allrecipes” or “Myrecipes”, and the H-nodes are sets of tags assigned to search engines, tags such as “Recipes”, “Reference”, “Shopping”, “Books”, or “Electronics”. The tag-hierarchy is displayed as a vertical list of both D-nodes and H-nodes, one node per line. A line listing a D-node, i.e. a search engine, contains the icon and name of the search engine; there are two of them in the Engines panel of Figure 3, listing the D-nodes “Allrecipes” and “Myrecipes”; all the other lines listing H-nodes. A line listing an H-node contains a triangle icon, which is sometimes omitted (invisible), a folder icon, the name of a tag belonging to the H-node, and the number of D-children of the H-node. Only one tag is shown even if the H-node is a set of multiple tags. The other tags in the set are those shown in *ancestor lines*. (A first line is said to be an ancestor of a second line if and only if it is above and less indented than the second line, and there is no line between the first and second line that is at least as indented as the first line.) For example the line showing the tag “Books” lists the H-node containing the tags “Shopping” and “Books”.

A triangle icon has two states. It may be “closed”, pointing to the right, or “open”, pointing down. For example the line showing the tag “Shopping” in Figure 3 has an open triangle icon, while the line showing the tag “Books” has a closed triangle icon. A folder icon also has two states, “closed” and “open”, indicated by self-explanatory graphics. For example, the line showing the tag “Recipes” in Figure 3 has an open folder icon, while all other folder icons in the figure are closed.

When a line listing an H-node has a closed triangle and a closed folder, and the user clicks on the closed triangle, Noflail Search changes the state of the triangle icon to “open” and inserts the H-children of the H-node under the line, indented to the right. (If the H-node has no H-children the triangle icon is omitted.) Figure 4 shows the state of the Engines panel of Figure 3 after the user clicks on the closed triangle in the “Books” line. The H-node of the “Books” line, which consists of the two tags “Shopping” and “Books” happens to have three H-children, which are the three H-nodes obtained by adding the tags “Electronics”, “Games” and “Media” respectively to the two tags “Shopping” and “Books”.

If the triangle and folder are both closed and the user clicks on the folder, Noflail Search changes the state of the folder icon to “open” and inserts the D-children of the U-node under the line, indented to the right. (An H-node always has D-children, since it consists of a set of co-occurring tags.) Figure 5 shows the state of the Engines panel of Figure 3 after the user clicks on the closed folder icon. As indicated on the “Books” line, the H-node of the line has three D-children, which are the search engines “Amazon Books”, “Half.com” and “Powell’s”.

Only one of the two icons in an H-node line can be open at any one time. If the triangle is open, the H-node line is followed by *descendant lines* (lines of which the H-node line is an ancestor line) listing the H-children of the H-node indented to the right, possibly interspersed with lines listing grand children and further descendants of the H-node. When the user clicks on the open triangle, Noflail Search changes the state of the triangle icon to “closed” and removes all the descendant lines.

The user may also click on a closed triangle icon when the folder icon in the same H-node line is open, or on a closed folder icon when the triangle icon in the same H-node line is visible and open.

When a line listing an H-node has a closed triangle icon and an open folder icon, and the user clicks on the closed triangle icon, Noflail Search performs the following four steps: it changes the triangle icon to the “open” state; it changes the folder icon to the “closed” state; it removes the lines listing the D-children of the H-node that follow the line that lists the H-node; and it inserts lines listing the H-children of the H-node, indented to the right.

Similarly, when a line listing an H-node has a closed folder icon and an open triangle icon, and the user clicks on the closed folder icon, Noflail Search performs the following four steps: it changes the folder icon to the “open” state; it changes the triangle icon to the “closed” state; it removes the descendant lines of the line that lists the H-node; and it inserts lines listing the D-children of the H-node, indented to the right.

Thus, when the Engines panel is as shown in Figure 5 and the user clicks on the closed triangle icon of the “Books” line, the Engines panel becomes as shown in Figure 4. And when the Engines panel is as shown in Figure 4 and the user clicks on the closed folder icon of the “Books” line, the Engines panel becomes as shown in Figure 5.

This method of browsing a tag hierarchy differs from the prior art methods as follows.

- It does not require navigation, by contrast with the first prior art method. Thus it does not impose on the user the burden of remembering the structure of the hierarchy.
- It displays H-children and D-children separately, by contrast with the first, second, third and fourth prior art methods, so that H-children are not be crowded out by too many D-children, nor D-children by too many H-children, particularly in a constrained space.
- It does not use more than one area of the display, by contrast with the second, fourth and fifth prior methods, and is therefore more suitable for use in a constrained space.