Intelligent Menus

ELEC-E7870 - Advanced Topics in User Interfaces
11.03.2019

Kashyap Todi & Sunjun Kim
Learning Objectives

- Learn about self-adapting menu layouts (SAM)
- Deploy SAM on a webpage of your choice
- Experiment with adaptation styles and policies
Self-Adapting Web Menus

github.com/aalto-ui/sam
Menus

- Item A
- Item B
- Item C
- Item D
- Item E

A list of items
Contains some ordering (positions)
Can contain groups and sub-groups
Menus

Typically static
Can contain several items & sub-groups
Increased visual search times
MenuOptimizer

[Bailly et. al, UIST 2013]
Self-Adapting Menus

Item A
Item B
Item C
Item D
Item E

Item C
Item B
Item D
Item A
Item E
SAM

User Logging → Metrics → Target Policy
Menu Abstraction → Menu → Adaptation Style

[Goebert et al., IUI 2019]
SAM

User Logging → Metrics → Target Policy

Menu Abstraction → Menus → Adaptation Style

Scores → DOM Modifications

Events → Item A → Item E

DOM → Item D → Item A

[Item B, Item C] → Item C → Item B

[Item A, Item E] → Item E → Item A

[Gobert et al., IUI 2019]
Abstract representation of the menu that can be interpreted by underlying algorithms

- Number of elements (items)
- Position of each element
- Grouping of elements
User Logging

- Mouse clicks on menu item
- Page visited (URL)
- Time of visit
- Duration
- etc.
Target Policies

Used to compute scores for each element

- Click Frequency
- Page Visit Frequency
- Total Page Visit Duration
- Page Visit Recency
- Serial-Position Curve
- AccessRank
- <Your own policy here>
Adaptation Style

Determines visual changes made to the menu

- Highlighting
- Item Reordering
- Group Reordering
- Folding
- <Your own style here>

Multiple styles can be combined to form a composite style
Usage
As an end-user

Step 1: Inject jQuery, sam.css, sam.js across websites
Step 2: Specify the tag for the menu on a target webpage
Step 3: Specify sub-menus in the menu
Step 4: Specify individual menu items’ tag
As a researcher

Experiment and add new adaptation styles

1. Implement the AdaptationStyle interface
2. Add a new custom CSS class to selected items
3. Specify desired CSS formatting

Read the paper for more info: kashyaptodi.com/sam
As a researcher

Experiment and add new policies

1. Extend the DefaultTargetPolicy interface
2. Implement `getSortedItemsWithScores(menuManager, dataManager)`
3. Return items with scores calculated by the policy

Read the paper for more info: kashyaptodi.com/sam
DEMO
Workshop Tasks
Workshop Tasks

1. Get started:
   1. Download the repository from github.com/aalto-ui/sam
   2. Install npm and node.js on your machine
   3. Install grunt via npm (npm install grunt-cli)
   4. Compile and build sam:
      1. npm install
      2. grunt
   5. Install CodeInjector extension for Firefox
   6. Inject jquery, sam.css, sam.js
   7. Inject .js files for example websites (from SAM repository)
   8. Try out SAM
2. Add a new webpage menu
   1. Select a webpage of your choice (should have a menu)
   2. Inspect the page source to find the menu description
   3. Create a `<mypage>.js` file for your page
   4. Specify the menu and menu items for SAM
   5. Inject the `.js` file into your browser to enable SAM

Done? Demo your webpage to the group!
Workshop Tasks

3. Experiment with styles and policies
   1. Navigate to sam/src/ts/adaptations.
      - Policies are under /policies
      - Adaptations styles are under /styles
   2. Edit existing, or create new, policies and/or styles
   3. Rebuild by using grunt
   4. Inject the new sam.css and sam.js files

Read the paper for more info: kashyaptodi.com/sam
Questions?