Crowdsourcing Interface for Collecting Correspondences of Web Pages

Juho Kim, Ranjitha Kumar, Scott R. Klemmer
Stanford University HCI Group
Computer Science Department
Stanford, CA 94305, USA
{juhokim, ranju}@stanford.edu, srk@cs.stanford.edu

ABSTRACT
One challenge in building a web design tool that attempts to leverage examples is gathering design alternatives and providing mappings between web page elements. We present a crowdsourcing interface to collect user-generated correspondences between two web pages. Our iterative refinement of the interface was guided by three main design principles: modularize the task, minimize user errors, and provide relevant information. As an initial experiment, we collected fifteen web pages with diverse style and layout, and deployed the interface on Amazon's Mechanical Turk. Preliminary data analysis shows that Turkers take longer than experts and define fewer mappings in general. Further evaluation and experiments with different types of pages will identify directions for a web design tool that enables the use of any web page as a design template.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Web design, crowdsourcing, Mechanical Turk

INTRODUCTION
Web designers experiment with a number of design alternatives in the early stages of a design process. Concrete, existing design artifacts facilitate creative design, and merging them is a common task in idea exploration [2]. Kumar *et al.* [3] introduced a web design tool that attempts to leverage examples by enabling the use of any web page as a design template. One challenge in building such tool is collecting examples that fit the current context and merging the examples with the working design. Could crowdsourcing be a viable solution in enumerating design alternatives? Moreover, could we crowdsource the task of mapping between two web pages to collect a diverse data set of correspondences?

We present a crowdsourcing interface that collects user-generated mappings between pairs of web pages. The interface provides direct, exploratory mapping interactions for visual web page segments. It contributes to the research goal of building a web design tool that automatically retargets the current content onto any web page layout [3]. The mapping results serve as the training data for an optimized machine learning algorithm responsible for automatic retargeting.

The value of crowdsourcing in this specific context is twofold. First, the crowdsourcing system offers scalability as the database of mapping results expands. Second, it is capable of rapidly collecting a large amount of data that covers a wide range in the design spectrum.

INTERFACE OVERVIEW
The interface juxtaposes a pair of web pages on the screen. Each page is internally decomposed into a hierarchical representation, based on conventional DOM (Document Object Model) structure and visual cues such as position, size, font, and color. We represent the hierarchy in the context of the original layout. The tree structure is not explicit; only parts of the tree are visible based on the current cursor position. The interface highlights with a green box a single segment on the left-hand page and asks the user to select a corresponding segment on the right-hand page. If the user decides that there is no appropriate mapping for the current segment, they click a 'no match' button. The process repeats until all segments on the left page are handled.

DESIGN DECISIONS
Kittur *et al.* [1] recommend crowdsourcing task creators to design verifiable questions. Our design decisions attempt to comply with this guideline by breaking down the complex mapping process into a number of simple micro-tasks. The

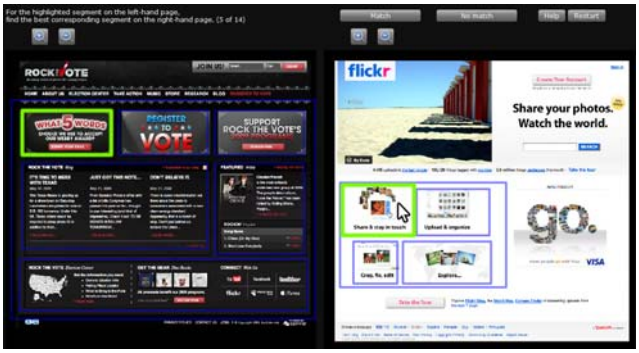

Figure 1: Overview of the crowdsourcing interface for collecting mappings between two web pages. Users mouse-over to explore page segments, and select and click 'Match' to add a mapping.

UIST'09, October 4–7, 2009, Victoria, BC, Canada.

tool requires users to explicitly state a corresponding node for each micro-task, and their answers can be vetted later for quality. Rather than selecting both left-hand and right-hand segments, users only need to pick a right-hand segment that best matches the highlighted left-hand node. Our pilot study showed that users completed the task faster when nodes were presented as a serial task rather than a free-form mapping task.

Users had difficulty precisely matching the regions, because segments are small in size and often cluttered. Our pilot subjects found it especially frustrating to select a parent segment when children segments cover the same space as the parent. Our solution to this problem was to bloat parent nodes so that there are always margins between parent and children for easier parent selection.

Observation of pilot users and their mapping data revealed that they take into account both visual cues and semantics of the page. We added a 30-second preview for each web page shown in the original size, allowing them to read through the content. Some participants complained that they could not plan ahead because there was no way to see how pages are segmented. We added visual cues so users could highlight locally-related nodes upon mouse-over on any node.

METHOD

We harvested fifteen different web pages for the mapping tasks. Diversity in style, layout, information density, and genre were main criteria in web page selection, and our intent was to see how people handle tasks with two dissimilar pages. We cut all pages with two different granularity parameters, so that the high-granularity version contains further subdivided segments that are not present in the low-granularity version. The interface displays the low-granularity pages on the left-hand to reduce the number of steps for each task, and the high-granularity on the right-hand to enable more fine-grained segment selection. The number of segments ranges from 6 to 24 for the left-hand (mean=15.1, std. dev.=5.6) and from 8 to 31 for the right-hand (mean=23.2, std. dev.=7.4). The system randomly selects a pair of web pages from the page database.

We ran two separate experiments on Amazon's Mechanical Turk. We collected 240 mapping entries in total from 80 Turkers. In the first experiment, each participant completed 5 distinct mapping tasks (40 Turkers, 200 entries) while in the second setup, only 1 task was assigned per Turker (40 Turkers, 40 entries). We added the two conditions to see if the number of tasks per Turker affects the task duration or the result quality.

Experiment	Authors	MTurk 1	MTurk 2
Mappings assigned per user	210	5	1
No. of users	2	40	40
Total mappings collected	420	200	40
Task duration (sec)	156.8	277.1	340.6
Nodes mapped (%)	91.5	71.2	71.4

Table 1: Summary of experiment setup and results.

RESULTS

To establish a baseline, two of the authors each mapped all 210 pairs to compare their data with crowdsourced mappings. While the authors only spent 156.8 seconds (std. dev.=138.9) per task, Turkers spent 277.1 seconds (std. dev.=140.1) per task when assigned 5 tasks and 340.6 seconds (std. dev.=141.6) when assigned 1 task. This suggests that Turkers were not gaming with the system, spending long enough on each task. It also implies a learning effect and that the interface or task could be simplified further to make learning easier.

To roughly evaluate the quality of the results, we calculated the percentage of 'no match' segments in the data set. Although an acceptable proportion of 'no match' segments would differ between pairs, too many 'no match' nodes suggest that the response is suspect. Authors defined mappings for 91.5% (std. dev.=14.0) of the nodes, while Turkers did for 71.2% (std. dev.=21.0) under the 5-task and 71.4% (std. dev.=21.5) under the 1-task condition. The crowdsourced results indicate that the number of tasks assigned had no effect in the quality of produced results.

FUTURE WORK

To assess the quality of the collected data, we plan to crowdsource the evaluation process by feeding results back into Mechanical Turk and having third-party raters evaluate mapping instances.

Future studies could collect pages that share common contents, goals, or design components to support the process of sensemaking for Turkers. Possible combinations include pages that have the same content targeted for different media types (mobile, desktop, print) and different versions of the same page over time. Each page group relates to specific use scenarios that could engage Turkers more.

The next step is to build the automatic retargeting tool, which presents a corpus of design alternatives produced by the machine learning algorithm and performs a synthesis of web page elements.

REFERENCES

1. A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with Mechanical Turk. In Proc. CHI'08, ACM Press, pages 453-456, 2008.
2. J. L. Kolodner and L. M. Willis. Case-based creative design. In AAAI Spring Symposium on AI and Creativity, 1993.
3. R. Kumar, J. Kim, and S. R. Klemmer. Automatic retargeting of web page content. In CHI'09 extended abstracts, Boston, USA, 2009. ACM.