PersaLog: Personalization of News Article Content

Eytan Adar[1], Carolyn Gearig[1], Ayshwarya Balasubramanian[2], Jessica Hullman[3]

[1]University of Michigan, Ann Arbor, MI
{eadar,ccgearig}@umich.edu

[2]Deloitte
Arlington, VA
ayshwaryabalasubramanian@gmail.edu

[3]University of Washington
Seattle, WA
jhullman@uw.edu

ABSTRACT

Content personalization—automatically modifying text and multimedia features *within* articles based on the reader's individual features—is evolving as a new form of journalism. Informed by constraints articulated through a survey of journalists, we have implemented PersaLog, a novel system for creating personalized content (e.g., text and interactive visualizations). Because crafting, and validating, personalized content can be challenging to scale across articles (unlike feed personalization), we offer a simple Domain Specific Language (DSL), and editing environment, to support this task. PersaLog is particularly designed to support the personalization of existing text and visualizations. Our work provides guidelines for personalization as well as a system that allows for both subtle and dramatic personalization-driven content changes. We validate PersaLog using case and lab studies.

ACM Classification Keywords

H.5.1 Information Interfaces and Presentation (e.g. HCI): Multimedia Information Systems

Author Keywords

Personalized Content, News Personalization, Guidelines

INTRODUCTION

Personalization and customization in journalism has a long history. Print and news broadcasters have long crafted localized editions in "native formats" (printed papers or TV shows). Providers like the *The New York Times* have offered printed editions based on a reader's location and, more recently, personalized homepages on the web. Aggregators, such as Google News, allow readers to customize their news feeds in a number of ways. Content producers benefit from this type of *feed personalization* as it allows them to leverage archival content and create local or hyper-local editions. Readers benefit from lessened information overload and increased engagement.

Content personalization—automated changes to the facts presented in a single news article according to inferable properties of the reader–has only begun to emerge as a viable feature. Through content personalization, a news site automatically customizes *text* and *multi-media* (e.g., visualizations) for a specific article. Automated content personalization allows a journalist to write one article that automatically customizes itself for many readers. As with *feed* personalization, *content* personalization, can increase engagement and learning (e.g., [13]); and supports behavioral change (e.g., [17]). Organizations with access to talented journalists, designers, and engineers, such as *The New York Times*, have begun to provide custom (one-off) personalized experiences. In May 2015 [2] and April 2016 [1], *The Times* published stories that personalized maps and text for each reader based on reader location (see Figure 1). Visualizations, including maps, charts, and tables were dynamically altered to focus on the reader's location (e.g., automated zooming on the map, local comparable counties in the table, etc.). Text modifications, similarly, personalize information (e.g., *local* poverty numbers, probability of overcoming poverty, etc.). Unfortunately, the rarity of such stories signals the difficulty in creating this kind of material.

Feed personalization has been much easier to adopt, in part, because a single algorithm can be applied across an entire site. The algorithm can combine inferences about the reader as well as prior behavior to make personalized article *suggestions*–a more traditional ranking problem. In contrast, content personalization requires significant, article-specific work. Multiple individuals in a news organization (e.g., authors, editors, copy-editors, fact-checkers, programmers, graphic designers) must contribute to build inference engines, personalization rules, and visualizations that can be modified to present personalized views. The effort is further complicated by the lack of design guidelines and tools that can be applied easily to new content.

In this paper we present PersaLog, a Domain Specific Language (DSL) and system for creating personalized news article content[1]. PersaLog supports personalization of both new content and existing, unpersonalized, information. In addition to allowing a journalist to personalize *generic* text content, PersaLog provides similar support for existing interactive visualizations. While direct-manipulation interaction (e.g., brush, filter, etc.) are common features, the event model they use is difficult to script–a key requirement for automatically putting them into a personalized state. PersaLog wraps existing visualizations, such as those built with D3, so that they can become part of the personalized content experience.

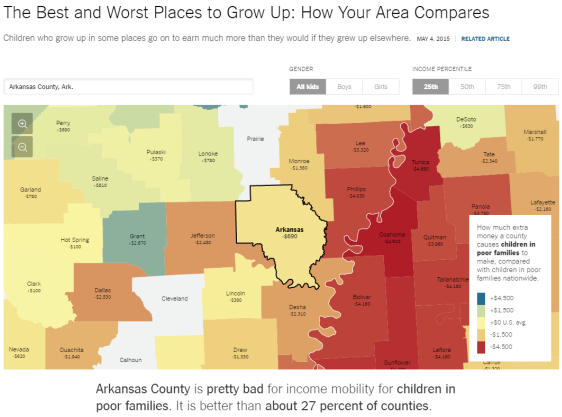
We define content personalization for news as: *"an automated change to the **set of facts** that appear in an **article's content** based on **properties of the reader**."* PersaLog's support for *user modelling* allows the journalist to identify *properties*

CHI 2017, May 06-11, 2017, Denver, CO, USA

 ISBN 978-1-4503-4655-9/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3025453.3025631

[1]We use different fonts when we want to distinguish between the system (PersaLog) and language (PERSALOG).



...

Consider Arkansas County, Ark., our best guess for where you might be reading this article. (Feel free to change to another place by selecting a new county on the map or using the search boxes throughout this page.)

It's below average in helping poor children up the income ladder. It ranks 668th out of 2,478 counties, better than about 27 percent of counties. It is relatively worse for poor girls than it is for poor boys.

Here are the estimates for how much 20 years of childhood in Arkansas County adds or takes away from a child's income (compared with an average county), along with the national percentile ranking for each.

...

Figure 1. Sample personalization of image and graphic in The Times' article "The Best and Worst Places to Grow Up: How Your Area Compares" (Simulated view generated from a computer in Dewitt, Arkansas which is in Arkansas County.)

of the reader: features and preferences of the reader/viewer from intrinsic (e.g., personality traits) to extrinsic (e.g., age), and from explicit (e.g., the name I provided when signing up) to inferred (e.g., movie interests inferred from from past search behaviors. PersaLog currently supports two common properties: geolocation and demographic inference.

Authors can personalize content–which includes text and multimedia features (e.g., charts, static or interactive, photographs, videos, etc.)–through PERSALOG or higher-level GUI tools. PersaLog supports modifying any web page element that can be programmatically changed, thus reconfiguring the *set* of facts in the article. *Facts* are the information-carrying 'atomic units' by which the journalist can personalize content (adding, removing, or emphasizing key facts). For example, "unemployment in county *Y* is *x%*" may only be an interesting fact to residents of *Y* or nearby locations, and could be replaced by more *locally* relevant information. In another article, "*Sobotka* is the prime minister of the Czech Republic" is a contextualizing fact valuable to readers outside the Czech Republic.

To determine the best design for PersaLog, we surveyed 22 media professionals and interviewed creators of personalized content from *The New York Times* and *Washington Post*, with the goal of understanding definitions, possibilities, and limitations of personalized content. Our work contributes a distillation of this information into a broader set of guidelines and design criteria for content personalization in news, which we utilize explicitly in architecting and building PersaLog. PersaLog is designed to serve the needs of journalists in *creating* personalized content to increase engagement and educational value. Conversely, the system simplifies deployment to *consumers*. We describe a number of innovations designed to support a broad set of personalization pipelines and demonstrate their applicability through a set of case studies and a controlled lab study to validate the usability of the DSL.

RELATED WORK

News Personalization

Feed personalization, or automated curation of articles based on information about an individual, was originally implemented to help people deal with information overload (e.g., [11]). Research has since expanded to support presentation of archival content, increasing engagement, and the creation of 'sub-properties' (e.g., local news through hyperlocalization) [59]. Collaborative filtering approaches have been broadly studied as mechanisms for filtering, sorting, or organizing a stream of news articles [11, 22]. Subsequent research has found that feed personalization can also increase engagement (e.g., [10]). News services including the *BBC*, *The New York Times*, *The Huffington Post* and *NPR* and aggregators (e.g., Google News) have integrated feed personalization.

Content personalization[2] is much less evident in journalism research and practice. A recent article by *The New York Times* (see Figure 1) is one highly visible example [2]. The article dynamically adjusts visualizations (e.g., a thematic map) and article text based on geolocation (inferred by IP address). In the context of broadcast news, the *BBC* has used *object-based broadcasting*–content 'chunking'–to shorten or otherwise personalize broadcasts while still retaining coherence [34]. Creating interactive features is time and resource consuming. Smaller publishers, without the resources of *The Times*, would be particularly constrained in their ability to produced personalized content given current technology.

Abstractly, any article with interactive content may appear to support personalization (i.e., is *adaptable* [49]). Take the example of an article that contains a map that users can modify based on ZIP code. Contrast this to the article that *automatically* infers (i.e., is *adaptive*) the ZIP code from the reader's IP address to adjust the visualization. In developing PersaLog, we focus on the latter definition of personalization. That is, if the *only* way an article's content is changed is in reaction to the reader's *direct manipulation*, we do not consider this personalization (though we acknowledge that such actions produce personally-relevant views).

Personalization in other domains

Personalization is often used outside of the news domain. For example, personalized advertising is fundamental to advertis-

[2]We note that *feed personalization* is sometimes also called *content personalization* in the media. We use the more specific meaning here.

ing practice and Internet-based ads (e.g., [51, 42]). Advertisers on Facebook can customize ads based on properties of location (e.g., whether a user lives there, is currently at a location or has been there recently) and other demographic data (e.g., age, gender, educational level and interests inferred from trends in a user's profile) [19]. Google supports locally-targeted results (e.g., a query for "pizza" will return different ads depending on location) [27]. In contrast to advertising, *news-driven* content personalization brings new challenges based on journalistic values such as transparency [55].

Content personalization is often effective in *learning environments* (e.g., [13]). Specific instances include environments that: tailor content based on learning-style [37, 50]; use fictional text to achieve learning objectives, such as by replacing generic terms with personally-relevant ones (e.g., favorite foods in word problems) [13, 16]; and support self-regulated learning [15, 43]. In medical applications, patient education (e.g., personalized pamphlets for behavioral change [56]) has also benefited from personalization (e.g., see [17, 26]). To make complex information more relatable, recent visualization tools automatically generate maps that show a user a personalized spatial analogy for a distance, area, or other spatial reference that she encounters in an article [36]. While the specific objectives and values of content creators in these domains are often different from news, they inspire our design.

Work in *adaptive hypermedia* and *education* ([18, 39]) led to sophisticated ways to modify hypertext-like structures or text by 'bundling' hypermedia objects based on higher-level design goals (e.g., education). These systems leverage a user model for personalization but often build experiences from the ground-up, rather than augmenting existing content. Hypertext DSLs (e.g., [6, 47]) inspire our design, but are generally too complex for end-users to learn or use (and tend to be evaluated for expressiveness rather than usability). *Natural Language Generation*, focuses on automatically generating unstructured text (e.g., [54]) from structured content (e.g., a database of animal characteristics). Like adaptive hypermedia, NLG can be personalized through a user model [46]. While our goal is not to fully automate the generation of personalized text, we can apply these ideas to improve PersaLog. For example, personalization may generate a large set of article variants [38] which may be difficult to copy-edit. Automated 'repair' features of NLG [31] can test and correct personalized text.

Adaptive interfaces that customize GUI layouts based on prior use or end-user properties are a form of personalization (e.g., [20, 23, 24, 60]). These are often built to improve productivity and usability by streamlining direct manipulation. Though they are different in intent from content personalization, some of the guidelines that emerge from our survey of journalists align with best practices in adaptive interface research at a high level. For example, a lack of user control is noted in the broader interface personalization literature [20, 32].

Research and applied work in End-User Programming (EUP) also provides some guidance in our development of PersaLog (see [40] for a survey). Various solutions provide DSLs to enable control over applications, devices, and–most closely to our domain–Web environments [14, 61]. While EUP often targets individual (i.e., 'code for one's self' [40]), rather than large-scale deployment, there are notable exceptions (e.g., CoScripter [41]). With PERSALOG we have adopted a 'little-language' approach [3]. In contrast to many DSLs, PERSALOG is neither a general programming language nor a library (e.g., [4]). As with the adaptive-hypertext environments [7, 21], we do not believe visual programming environments (e.g. [63]) will naturally fit into the journalist habitats (i.e., CMS systems [57, 62]). However, this may be worth pursuing in future research.

Superficially, PersaLog is related to web-templating languages such as PHP, Mustache, Jinja2, etc. While elements of PersaLog can be integrated into these tools, the PERSALOG language was (a) developed to support personalization tasks, and (b) designed to more easily modify *existing* unpersonalized content (including visualizations) to enable personalization.

CURRENT EXAMPLES

We have identified three recent examples of content personalization from *The New York Times* [1, 2, 53] and one from the *Washington Post* [45]. We summarize key implementation details and our interviews with their authors (Kevin Quealy, Graphics Editor at *The Times* [52] and Darla Cameron and Denise Lu, Graphics Editors at *The Washington Post* [9]).

The focus of all four articles was geography and we found four mechanisms for personalization in text: *"You are here" (YaH)* which declare where the article thinks the reader is (e.g., "...rethink what they know about health costs in **Arlington, VA** ..."); *comparative with data (CD)* where local places are related to others with numerical data (e.g., "It ranks **175th** out of 2,478 counties"); *relative comparison (RC)* (e.g., "If you're poor and live in the **New York** area, it's better to be in **Putnam county** ..."); and *local statistics (LS)* (e.g., "...researchers examined prices at **7** hospitals in **Arlington, VA** ...founds surgery can cost ... **$22,800** ..."). Personalized visualizations consisted of maps, bar charts, scatter plots and tables.

We identified the following in the articles: [1] personalized 7 of 14 paragraphs (3 YaH, 2 CD, 1 RC, 1 LS), 1 map and 2 tables; [2] personalized 9 of 18 (3 YaH, 3 CD, 1 RC, 2 LS), 1 map, 1 chart, and 3 tables, this article also allowed interaction to focus on different genders or income percentile which further changed the text; [53] personalized 4 of 38 paragraphs (2 YaH, 1 RC, 2 LS), and 3 of 6 visualizations (a dot plot, scatter plot, and bar chart); [45] personalized 1 of 20 paragraphs (LS), 1 of 9 maps and 1 of 2 charts.

In terms of implementing geolocation, *The Times* leverages an existing server-based solution within *The Times* organization. Mr. Quealy indicated that this greatly simplified his work. It is notable that a smaller organization may not have this capability. The *Post* utilized the HTML5 Geolocation service–an opt-in, browser based solution. This makes the the personalization obvious (a feature recognized by the *Post* reporters). However, it is disruptive (a pop-up requests location access) and will not work on all browsers. Our interviews all emphasized resource constraints (time and developer access) as major reasons why new personalized pages are difficult to create. Specifically, that a great deal of care was necessary to ensure that the content "made sense" and worked consistently (e.g., failure modes if

there was no data). Grammatical correctness was critical. For example, correct pluralization, comparative language, and in the case of the wealth article [2], gender and income (e.g., "richest girls" vs. "poorest boys").

FORMATIVE SURVEY OF JOURNALISTS

To inform our development of PersaLog, we conducted a survey of journalism professionals. We were interested in how journalistic values, such as transparent reporting, impact personalization. We also wanted to understand the specific workflows (editorial decision-making, deadlines, etc.), resources (data and personnel), constraints, and possible barriers to adoption. Our survey broadly targeted: 1) how journalists currently understand personalization, 2) the perceived benefits and risks of content personalization, 3) possible dimensions (e.g., location, age, political orientation, etc.) for content personalization, and 4) attitudes towards adoption in a news organization. The survey captured basic information such as the journalist's specific role and content responsibilities in the news organization. We circulated the survey by sending it to our contacts in journalism, including professionals at national and local publications, through Facebook groups for journalists, and through Twitter. We did not target data journalists or those expected to have strong programming backgrounds.

Twenty-two journalists completed our survey, including: 6 reporters, 3 graphic developers/designers, and 6 with other editorial roles (e.g., visual editor for a large national newspaper, editor for experimental mobile projects at a national newspaper). Seven other respondents held unique positions including: digital product analyst (national paper), audience development editor (national magazine), and a videographer (university newspaper). Fifty percent of respondents worked in journalism for 1-3 years, 27% for 4-6 years, and 14% worked over 6 years. Six work directly with graphic production and editing, and another 14 with graphics in some fashion (e.g., content editors who manage overall story presentation, text and visuals). Eleven participants answered a "programming inventory." Of these, most (10) reported experience with Excel, Google Spreadsheets, and all (11) with HTML/CSS. Four (of 11) considered themselves highly competent at JavaScript[3].

Responses

When asked "What does content personalization mean to you?" (before being provided with a definition) the majority of journalists we surveyed (16 out of 22) described feed or homepage personalization. Only 6 described content personalization as content adapted to reader properties. As one graphics developer (P20) put it, content personalization is "[M]aking a national story mean something specific and (hopefully) different for every reader based on their background." Others mentioned specific reader properties in their definitions: "Targeting of content based on either imputed (IP or other geographic coding) or provided (registration-related) information" (P3). After we defined content personalization and provided examples, our respondents identified features on which they could imagine personalizing. Location was the most frequently mentioned property for personalizing content (16 respondents), followed by age (12), prior internet behavior (9), gender (5), political preferences (4), and race (3).

Our respondents described various high-level goals for personalization, including increasing relevancy of the content (5), making the news more relatable or engaging (5), increasing memorability (3), prompting users to read the article in the first place (3), gradually introducing complex data (1), more effective communication (1), improving news comprehension (1), and increasing length of visit, recirculation, and brand loyalty (1). Several respondents expressed the importance of considering how the specific personalized content adds to the article. As one respondent (P3) described, content personalization is "an editorial decision like any other. Sometimes, the national context is important and the local data is meaningless noise. Sometimes, the opposite is true . . . [Y]ou have to decide all of this on a case-by-case basis." Privacy and bias issues, as we describe below, were noted by respondents and influence our design. At the high level, some of the guidelines we identify below are common to all systems with personalization (e.g., bias). However, the relative weight of each is unique to content personalization in the news domain.

Guideline 1: Anticipate potential bias—Content personalization is likely to occur per-article, rather than uniformly applied to an entire site. This lessens the possibility of algorithmic curation introduced biases known as "filter bubbles" or "echo chambers" (e.g., [5]). However, 8 respondents were concerned that bias may emerge, for example through the continuous use of the *same* personalization features in the *same* way (e.g., only providing local unemployment information in every article about unemployment). One journalist (P16) noted that, "if something ... isn't as 'bad' in a certain reader's area, they may disregard an issue, like income inequality." Another (P15) likened personalization to "hiding information from [readers] because your algorithm thinks it is irrelevant . . . , then they might leave . . . with an unintended conclusion."

Guideline 2: Provide reader control—Multiple journalists expressed interest in using content personalization provided that readers were given some control. This notion is related to *scrutability* (the ability to inspect the model) in personalized learning systems [35]. If, "the personalization gave readers the option to view others' personalizations, I would use it," one said (P8), and ". . . personalization is terrific in speaking directly to readers as long as there is still broader context for the information being presented or the option for the reader to see the general text/graphics," said another (P10). Journalists described how the ability to disable personalization may satisfy (some) privacy concerns, allow for a common view for discussion, and provide additional context to reduce bias.

Guideline 3: Signal personalization—To support reader control, a personalized interface should convey to the reader that content has been altered. The need to ensure that a reader knows *how and where* an article's text and images has been personalized may lead to the development of consistent visual cues (e.g., we utilize double underlines).

Guideline 4: Consider inference quality–Several journalists expressed concern with, what they perceived, as the low

[3]these numbers may be biased as journalists without programming experience may have ignored this aspect of the survey.

precision of automated inference. Journalists are well aware of the 'horror stories' of geolocation and data mining gone awry [30]. These respondents seemed to distrust implicitly inferred information. One journalist (P6) stated, "I would be hesitant to make a claim about a city based on raw data and not actually reporting from the ground or talking to sources." Ideally, a personalization system could incorporate inference confidence into display decision. For example, if the precision of an inferred property like socioeconomic status is expected to be low for certain regions, the system would not personalize.

Guideline 5: Sensitivity to reader privacy—Seven of our respondents remarked on the potential of personalization to inspire concerns about privacy. "Requesting personal information can be a big turn off for users and many will choose not to interact with your application at all," said an interaction designer (national paper) about graphics that request reader information (P19). She described her organization's existing attitude on personal information: "It's important to make users feel like their information is protected." This range of responses was also evident in reader reactions to *The Times'* personalized articles [58]. Additional research is necessary to determine *reader attitudes* around privacy for personalized content, but we suspect that an emphasis on sourcing materials and provenance may partially assuage concerns (e.g., by providing transparency on inference mechanisms and the availability of the data outside of the interface).

Guideline 6: Minimize application complexity for creators—A frequent theme in journalists' responses was that content personalization was desirable but would require more resources than their organization could easily provide. The limited technological capabilities of some journalists were one concern. Journalism workflows for a single article can involve many stakeholders with varying levels of technical expertise, content personalization tools should be accessible to users with limited programming background. From our interviews (with Quealy, Cameron and Lu), this was by far the most salient concern. All three described the difficulty in ensuring that language of the article worked (e.g., comparative language, unit pluralization, capitalization, etc.) and that an overall high-quality user-experience was maintained. An explicit design decision for PersaLog is not to provide complete automation in the *construction* of personalized content. Total automation is not required, or desired, as producing accurate, readable, grammatical content is placed at a premium.

Guideline 7: Design for scale—Respondents voiced concerns over manual effort for personalization: "The more you personalize (the stricter) the more data you have to present, more content you have to write" (P10). To scale personalization without requiring significant effort on a per-story basis, PersaLog should be designed with 'reuse' in mind. Mr. Quealy, for example, reflected on *The Times'* advantage in being able to leverage existing geolocation infrastructure. Our goal is to provide those with less resources with similar capabilities by creating a robust and reusable codebase.

PERSALOG ARCHITECTURE

We have created PersaLog, a JavaScript-based infrastructure and DSL, to facilitate content personalization. PersaLog is comprised of a simple set of tools that can be applied to existing content and which adhere to the guidelines outlined by our survey (Figure 2 summarizes the architecture). End-users create web pages with embedded PERSALOG code, either directly in HTML, or by using one of our editing tools (Fig. 2a). Visualizations and other graphical elements can also be integrated. Because most existing visualizations do not offer personalization features, PersaLog programmatically 'wraps' these to allow the journalist to automate direct manipulation interactions that were available in the original visualization (e.g., brushing, clicking, form filling). The *unpersonalized* content is transformed by PersaLog by translating PERSALOG code directly into JavaScript (i.e., transpiling). The standardization of the architecture enables experienced developers to add, or integrate, new personalization features into PersaLog. We describe each feature at a high-level below but point to `http://persalog.news` for more details and live examples.

User-modelling (e) occurs when a reader accesses a PersaLog-enabled page. Leveraging various implemented classifiers (f), PersaLog automatically attempts to build a user model (g). Services can be both internal and external to the PersaLog infrastructure. In the simplest case, the end-user's IP address is used to geolocate the individual. Other classifiers can predict demographics based on location, past use patterns or database values (for 'registered' end-users). Consistent with Guideline 2, any inference can be overriden by the reader. The generated user model, together with the compiled page, are passed to a *display reconfiguration* engine that re-renders the page. Automatically generated widgets allow the reader to disable personalization or change 'perspective.' Guidelines 1 and 2, anticipating bias and providing reader control, strongly informed the design of our reader-driven reconfigurability.

To test PERSALOG code, we have constructed a number of test harnesses (i) that can simulate different browsers and inputs (e.g., testing access from all unique zip codes). Depending on application, pieces of the architecture can be shifted easily from the client to server (e.g., Node.js). For example, pages can be personalized *on the server* before being passed to the client (this may have computational advantages). Alternatively, execution of the PersaLog can be done entirely in the browser. This off-loads work and supports privacy policies where high end-user control is preferred (Guideline 5). Our default instantiation of PersaLog is for compilation to happen at the browser, but user modelling to be handled on the server.

PersaLog Constructs: If-Segment-Then-Do

PERSALOG has two main constructs: *segment* and *do*. We use the term 'segment' based on its common usage in advertising. Both the user modelling components and rendering infrastructure utilize a pipeline of JavaScript functions. Each function accept a *person* object as input and an optional target (for rendering). When called, the functions assign one or more segment labels to the person and execute some action based on the value (e.g., gender, age; add sentences about that gender and age profile to text) and then pass the 'person' to the next function (PersaLog maintains a sequence of these functions and executes them serially). An analogous dataflow architec-

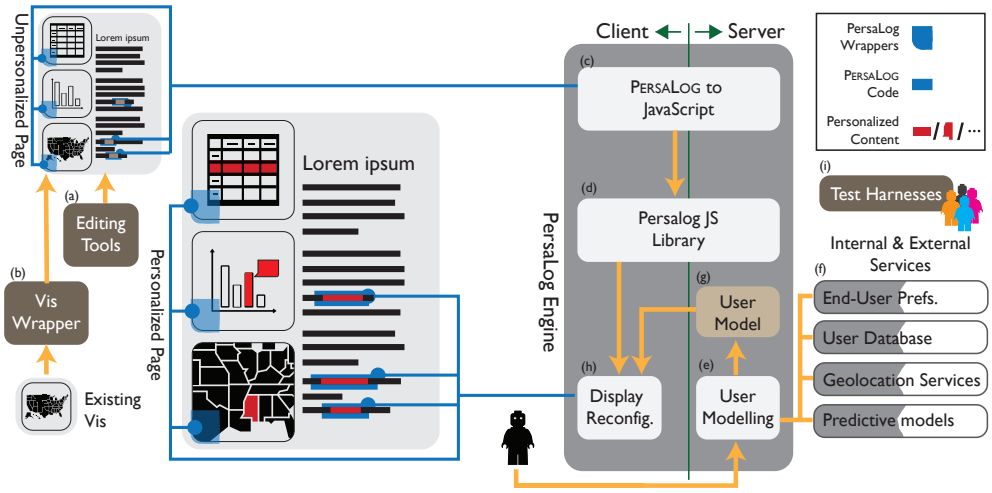


Figure 2. PersaLog Architecture

ture is the 'if-this-then-that' (IFTTT) style of programming: *if the person belongs to some segment, do something*.

Segmenting maps the person to a "type." This may be based on inference (i.e., given your surfing behavior I would put you in the 'teenager' segment or given the IP address, the reader would be 'segmented' by city, state and country), or based on explicit groupings or knowledge (e.g., based on the age provided by the user on sign-up, I place them into standard categories: 15-24, 25-44, 45-64, 65+, etc.). At the extreme is segmenting by unique visitor, which essentially creates a unique page for every individual. Regardless of the mechanism, the act of segmenting is equivalent to mapping a domain (i.e., all readers) into a range (i.e., segments). In PersaLog this is achieved either by using built-in code or by defining the functions using the PERSALOG language.

Using PersaLog to Create Personalized Content

PERSALOG *Code Blocks*

The rendering infrastructure for PersaLog works by: (1) identifying PERSALOG code blocks in a document; (2) generating "placeholder" HTML span in the text; and (3) interpreting the PERSALOG to determine what to place inside these dividers.

PERSALOG code blocks are hidden in HTML comment tags. This ensures that if PersaLog fails to execute, valid, readable content will still be generated. For example:

```
<div> Hello viewer. <!-- PersaLog
   ... PersaLog code here...
--></div>
```

During a document setup phase, PersaLog will add a new *span* inside of the *div* element, initially set to empty but with a unique id that is linked to the PERSALOG code in that block:

```
<div> Hello viewer. <span id="personal-1"/>
<!-- PersaLog  ...  --> </div>
```

If the PERSALOG code generates a string referring to the end-user's county, upon execution, the new HTML would be:

```
<div> Hello viewer. <span id="personal-1">
You seem to live in Livingston County</span>
<!-- PersaLog  ...  --> </div>
```

A PERSALOG code block can also refer to an existing HTML element (uniquely identified). In this case, output will be rendered into the matching DOM element rather than a newly created one. The original PERSALOG code remains embedded in the HTML and allowing the reader to inspect the personalization logic if they are interested (Guideline 3).

Personalizing Visualizations

Interactivity in visualizations is often driven by user actions (e.g., *mouseover* or *clicks*). Our goal for PersaLog is to ensure that these actions *can* be executed through code for two reasons. First, an interactive visualization, even one not intended for personalization, can often be personalized based on direct interactions. For example, a thematic map showing unemployment in the US may bring up key statistics by mousing over a particular state. If our goal is to personalize based on this action, we must simulate this mouse event. Second, we want to ensure that personalization doesn't create bias by limiting access (Guideline 1). A viewer that has disabled personalization should be able to use the standard interactive features to arrive at personalized content (manually). Thus, we should not execute actions that are impossible for a human to reproduce.

A special JavaScript file (*wrapper.js*) can be added to any existing page with interactive content or visualizations. The code operates on the content in two ways. First, all interactive HTML content (e.g., form elements) are identified and 'named.' Second, visualization content (currently only D3) is analyzed. We use a technique similar to that of the D3 Deconstructor [29] to 'parse' a visualization back into tabular representations. Roughly, each graphical 'mark' (e.g., rectangles, ellipses, or any other SVG shape) are isolated and connected back to the original data source. The unique identifiers in the data become the "names" (this allows us to say things like "click, California" instead of "click, 4th rectangle"). Specific mouse actions can be applied to visual elements that map to specific data rows. Internally, when invoking 'mouseover, California,' PersaLog will find the correct row, find the matching SVG element, and trigger the specific event. While this technique requires that columns with unique names (preferably,

human readable ones) in the original data tables, we have found this to be a reasonable assumption. Wrapped content can be reused in different contexts. PersaLog supports this by extracting the visual elements and displaying them in an embedded iframe (thus supporting reuse).

PERSALOG Syntax and Implementation

Below we describe key elements of PERSALOG syntax and execution. A sample block of PERSALOG code is:

```
if {{age}} <= 18 ::: print a
if {{age}} <= 65 ::: print b
default ::: print c
```

Variables are enclosed in double curly brackets (i.e., *{{foo}}*). Bracketing of this type, including double curly braces, was selected as it is common in HTML templating systems (e.g., AngularJS). All variables are bound to the *person* (a 'dictionary' object defined globally that holds all user model keys). Thus, *{{age}}* translates to *person['age']*. Evaluation of a PERSALOG code block proceeds from top to bottom. If all the conditions for a line are met, the person is placed in a virtual segment and the 'do' portion of the code is executed (in this case printing a single character) and the block is considered finished. If conditions on any line fail, the next line is executed. The PersaLog parser transforms this code into the following JavaScript block[4]:

```
function persalogBlock1_3(p,t) {
    if (p['age'] <= 18) {
        setSegment(p,'1_3','0');
        t.innerHTML = 'a';
    } else if (p['age'] <= 65) { ...
        t.innerHTML = 'b';
    } else { ...
        t.innerHTML = 'c';
    }
}
```

Each PERSALOG code block is transformed into a single JavaScript function, and each function is executed in turn during personalization (the person model and 'print target' are passed as arguments). The *setSegment* function assigns the person to a segment (in this case a 'virtual' one that PersaLog automatically created). *If* functions in PERSALOG can have a boolean form (e.g., *({{age}} > 10) && ({{age}} <= 20)* or *{{city}} endsWith 's'*–which is useful for generating grammatical correct pluralization when printing).

In some situations it is preferable to segment using a "fuzzy" match: that is, the closest match given a set of segments. PERSALOG supports a segmentation operator (*segmentby*) that is equivalent to a SQL JOIN but with more sophisticated criteria. The construction assumes the existence of a table (loaded with the *loadsegment* operator, one of the few exceptions to the segment-do language):

```
file: bigcities.csv

    big_city,    geoloc,         population
    New York,    "40.7,-74.0",   8.4M
    Chicago,     "41.8,-87.6",   2.7M
    Los Angeles, "34.0,-118.2",  10M
```

[4]This is a modified listing, additional code is added to bullet-proof the execution with try/catch statements. Also note that each block is uniquely identified

This comma-separated file is loaded and a new *segment database* called *bigcities* is loaded into the PersaLog namespace. Column names can be arbitrary but columns that start with *geoloc* are treated as locations and parsed into latitude/-longitude pairs. We can then create code such as:

```
loadsegment bigcities.csv

segmentby {{location}} nearestGeo
    {{bigcities.city}} setas
    {{closest}} ::: print you're
    closest to {{closest}} which
    has {{population}} residents.
default ::: donothing
```

Abstractly, one can think of *segmentby* as a large if/else that finds an optimal matching segment based on (potentially fuzzy) criteria. Rather than returning a boolean, as in the case of the *if* style construction, *segmentby* will return the best match. In the case above, the *nearestGeo* command cycles through each segment, finding the segment that best matches–which, in this case, is geographically nearest to our reader. The matching segment name is put into a *closest* variable (an optional criteria). The table columns for the matching segment (i.e., row) become part of that person's profile (e.g., *population*). This allows the PersaLog user to specify actions like *print {{population}}*. Other segmentation rules include: exact matches (e.g., *{{city}} matches {{bigcities.big_city}}*)–only people who live in one of the three cities will match); and nearest numerically. New JavaScript functions can be added to the core library and called in the same way. Such functions must simply produce one of the rows (or -1) and a 'distance' score. 'Distance' is semantically meaningful. In the geographical case it represents miles and can be used for further filtering (e.g., *{{closest_distance}}* $<= 50$ to require that reader be less than 50 miles away from some point).

Do-style operations include *print* (any HTML content can be produced here), *set* (set variables in the person's profile), *donothing* (a "null op"), and *vis*. Vis operations take as argument a function name and information to pass. For example, a PersaLog-enabled map might feature a *centerOn(location)* function which we could invoke as '*vis centerOn location*'. Visualizations 'wrapped' by PersaLog automatically have a set of functions they expose in this way (e.g., *mouseover*, *submit*, *click*, and other standard direct-manipulation events).

Any number of *segmenting* and *do* operations can be placed on a line: $c_0 ::: c_1 ::: \ldots ::: c_n$. When translating statements like this, PersaLog will find all *segmenting* functions (*do* or *segmentby*) and create a conjunctive query. For example, if c_1 and c_4 create segments, the translation would be: *if* $(c_1 \wedge c_4)$ *{ do* $c_0, c_2, c_3, c_5, \ldots\}$.

User Modelling

Like segmenting functions, user modelling functions accept a 'person' profile object as input and will *enhance* the profile through annotation. User modelling can take place within the browser or by using a remote server. The decision of where these services are housed can be based on the privacy policy (Guideline 5) as well as computational and memory constraints. For example, PersaLog currently offers geocoding services (IP address $\rightarrow$ geographical location). The database

Figure 3. Examples of the editor view for the article and code. Personalized maps (upper right) are shown for this article. One for nearby residents (localized) and one for remote readers who see a map relating the fire to a known location (Phoenix, AZ). The bottom right figure is an example of a personalization indicator.

housing this information is fairly large and can not reasonably be transmitted to every requesting client.

Currently, we support geolocation, an imperfect but widely used form of inference [33], through data offered by the MaxMind GeoIP service (**www.maxmind.com**). This produces latitude/longitude information based on IP address (from which zip codes, city, county, and state identification can be extracted). Once a location has been inferred, other location-specific features (e.g., *time zone*) are added. We have also built a simplified demographic inference engine based census data (specifically the American Community Survey) to support inference of demographic properties such as gender, age, and occupation based on physical location. This type of "weak" inference provides a maximum likelihood estimate based on the prevailing category. For example, we might infer that a person is more likely female because the particular city is 51% female. In keeping with Guideline 4–consider inference quality–PersaLog can report both a confidence in a particular inference as well as probability distributions over possible segments. Distributions are represented as probability density functions (PDFs) over the possible classes (e.g., probability of male versus female versus unknown, or probability of age$= 1,2,3,\ldots,100$. End users can use this information in deciding whether, and how, to use the inference. PERSALOG offers a special notation for this: $\{\{variable|PDF_{filter}\}\}$, where PDF_{filter} is area under the curve (AUC) of the PDF for a particular variable, v, given the filter. The filter may include ranges (e.g., $14--18$ or >21) and categorical filters (e.g., 'Engineer,Doctor'). A special *mostlikely* keyword can be combined with the *segmentby*:

```
segmentby mostlikely {{age|<18}} {{age|18-45}}
    {{age|>45}} setas {{agebin}}
```

This will segment based on *age* into one of three categories (based on which is mostly likely given the max AUC). For example, if distribution shows the biggest 'mass' under the range of 18–45, the *agebin* variable will be set to *18-45*. We do not anticipate this feature to be used by novice end-users, but provide the functions for more sophisticated applications where inference is highly noisy.

Additional user modelling engines (and more sophisticated user-models, e.g., [8]) can be added depending on the application. For services with 'ground-truth' classification, classifiers can be trained to infer properties as distinct as age, gender, political affiliation, and reading level from browsing or search behavior [12, 25, 48]. We emphasize that it is not the goal of the current prototype to innovate in inference algorithms but rather to support their integration. PersaLog also supports the use of non-inferred (i.e., explicit) user information. For example, if the site supports client registration with demographic information, PersaLog can obtain and use this information. Finally, PersaLog supports an override of user models (an end-user can change their model) and disabling of personalization, aligning with Guideline 2. Both inferred and explicit user attributes can persist beyond a specific page through cookies or as URL arguments (both are currently supported). Note that inferences can build upon each other: IP address can lead to location which can lead to demographic properties. The "inference pathway" (a directed acyclic graph) is maintained by PersaLog so that an override earlier in the pathway (e.g., to location) will optionally (but by default) "invalidate" the subsequent inferences, which are then recalculated.

UIs for Editing & Viewing

Creators of PersaLog-aware code can use the editor of their choice. We believe that using pseudo-code or markup languages while editing an article is familiar to journalists and many CMS's support this kind of writing (e.g., [62] and the New York Times' ArchieML [57]). Furthermore, our respondents indicated a broad knowledge of tools such as Excel and Google Spreadsheets, which have DSLs for manipulating data. For journalists comfortable with this level of programming, we have constructed a large sample database of PersaLog code (Guideline 7). That said, we recognize that that not all journalists will want to modify this code directly. Thus we have constructed additional interfaces including a "spreadsheet" style interface and a "wizard" to generate PersaLog code at a much higher level (Guideline 6).

Content creators can edit content in a modified version of the open-source Medium blogging system's editor [44]. A

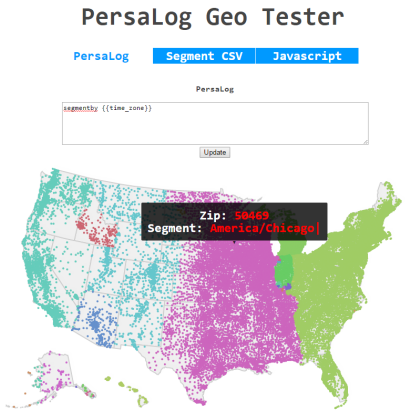


Figure 4. Visual test system–each point on the map is a simulated user, segmented by the PERSALOG code (and automatically colored)

menu for inserting images or formatting headings has been extended to insert PERSALOG code (see Figure 3, at left). A dialog box containing a two-column table is opened (Figure 3, middle). Each row in the table is inserted as a PERSALOG string (i.e., *A ::: B*–roughly the if/then structure, though the second column is optional to support *segmentby* and other constructs). A pencil icon (✎) indicates where this code is embedded; clicking brings up the editing dialog. The editor can export the page, with PERSALOG code, once finished.

By default we provide a number of features to: (a) indicate that personalization is happening (a fixed page footer), (b) that it can be changed (a dialog box is opened, allowing the reader to change their 'model'), and (c) that it can be disabled. Additionally, when mousing-over personalized content, a tooltip displays which "features" (e.g., age, location, etc.) were used to generate the content. This is done automatically by mining the code blocks. These feature are built to support Guidelines 2 and 3 concerning transparency and control.

Test Harnesses

We have created multiple test systems to allow writers to experiment and test PERSALOG code (Guideline 6), and which also support viewing how different readers will see the rendered page to gain awareness of potential biases (Guideline 1). These tools create simulated end-users (e.g., 30k with unique IP addresses corresponding to zip codes) that can be 'pushed' through a PersaLog personalization pipeline. Figure 4, a *map test harness*, is a visualization of these individuals plotted on a map. Different PERSALOG code will result in different colors based on segmentation. A similar page allows for testing of text and visualization 'blocks' by iterating over the simulated user sets. The generated JavaScript code is displayed and the 'object' corresponding to different people can be modified and sent through the pipeline. A third harness generates the text output for a simulated user base, identifies errors, and groups results (useful, for example, for validating pluralization).

EVALUATION

In addition to our survey guidelines, PersaLog and its specific features have been informed by feedback we received from journalists and through our interviews. To validate that PersaLog provides coverage of desired features we have constructed a number of case studies and a lab study.

PersaLog Understanding, Modification, and Construction

To evaluate the PERSALOG language among potential users we ran a study with 11 upper-level undergraduates working at a large university's newspaper. All had created news content but few had coding experience and none programmed regularly (four had taken introductory Python or C++ courses which are broadly required by many majors). Participants were given a printed tutorial explaining PERSALOG followed by a paper-based test consisting of 11 questions (see supplementary material) with three question types. Eight *interpretation* questions focused on describing the output of PERSALOG code (presented with the profile of the viewer, and using both complex and simple forms of segmentation), one *code modification* task asked the participant to extend code, and two *creation* problems had participant create code from scratch using different segmentation rules (traditional if/else as well as "segementby"). The tutorial provided sample code but no sample specifically answered the questions. Participants received $20. Median quiz time was 18 minutes (max: 23).

The median correct score was 8 (of 11). We labeled each incorrect answer as either a syntax error (e.g., used a single curly brace instead of doubles) or a more significant error (the mistake would not produce the correct result even if the syntax error were corrected). The median syntax errors was 1 (mean 1.45) and other errors was 2 (mean 2.27). While most questions were answered correctly by most participants, two were often incorrect. One was a piece of "buggy" code where the order of 'if' operations on a range were reversed (*if* $x < 22 \ldots$; *if* $x < 18 \ldots$ what happens when $x = 17$?). The second required creating a complicated segmentation followed by a second code if/else codeblock (none of the examples in the tutorial used this structure). Of the remaining questions, four were answered correctly by 8-9 students, and five were answered correctly by 10-11.

We designed this evaluation to focus on the PERSALOG language rather than the tools we created. This represents a 'worst-case:' students were largely untrained with PERSALOG aside from the tutorial, completed the task on paper (no explicit 'debugging support' or 'test-and-check' style programming), and were provided only a limited set of examples. Our goal was to determine language usability rather than identifying strengths and weaknesses in the "development environment."

Case Studies

We describe a set of case studies for stress testing PersaLog components using a diverse visualizations that show broad coverage of 'wrapping' and demonstrate expressiveness.

'How Your Area Compares'

We reverse engineered the code *The Times'* article [2], "The Best and Worst Places to Grow Up: How Your Area Compares," (Figure 1) to identify those places where personalization happened. While many personalized elements are straightforward (e.g., imputing user location and presenting relevant data), a few specific outliers exist (e.g., specific code around New

York and Washington DC). Once we identified these, we wrote PERSALOG code to generate this content. For all textual and visual elements, we were able to generate analogous content using PERSALOG code. This exercise also allowed us to test critical features. For example, we used string tests to ensure that pluralizations were grammatically correct. As part of this exercise we built our own generic visualizations (e.g., count-level thematic maps) which we 'wrapped.'

It is difficult to compare the PERSALOG-based version directly to *The Times* based on line-count. On the one hand, PersaLog does not require any complex operations to select DOM elements and offers a consistent inference library and has a simplified language for choosing what to display. However, the code we studied from *The Times* was obfuscated, compressed, and clearly made to be efficient (e.g., using shorthand: *if ? then : else*). We believe PERSALOG balances conciseness and readability for a broader population. The code for visualizations would largely be equivalent between the PersaLog and original versions. However, because PersaLog can wrap existing visualizations, we can leverage existing interactive assets and make them 'personalization compatible.' A final interesting difference is that the news publications' JavaScript often incorporated data-processing (e.g., transforming .234 to 23%). PersaLog does not remove the burden of generating or cleaning data. For example, the original article has a database of neighboring counties that are richer or poorer than your own. Currently, PersaLog treats this as a pre-processing step.

Visualization Tests

In addition to many simple D3 visualizations, we have also tested PersaLog within the context of more complex interactive visualizations. For example:

- Jim Vallandingham's "How Much Money do the Movies We Love Make?" (`http://vallandingham.me/vis/movie/`), an interactive visualization that allows the viewer to find movies by genre and plot them on a *rating* versus *profit* scatter plot. The visualization contained a number of brushing and search interactions, none of which were 'scriptable' originally. The PersaLog wrapper bound names to different interface and visual objects allowing us to perform actions like "select genre Action" and "mouseover film Iron Man." We have demonstrated personalization of this page based on reader age, setting the initial genre based on this inference.
- The Guardian's "Is Barack Obama the President?" (`http://www.isbarackobamathepresident.com/`), a whimsical visualization using balloons to represent states in the Obama–Romney election (2012). Hovering over a balloon provides more election information. We automatically invoke a hover over the viewer's home state. Again, the visualization was not originally designed to be controlled in this way and there were no "hooks" to the Texas object, for example.

Additional Examples

We have also created personalized articles that respond to age (indicating a reading level), by changing images and text to explain ion channels to an K-8 reader versus someone more advanced. Mr. Quealy [52] indicated that often this varying detail was important with technical material, with existing solutions largely composed of having 'expert explanations' pulled out as additional material (e.g., around topics like credit default swaps). We also built a hyperlocal personalization that changes maps based on location. An article about a forest fire shows either a broader map (if the reader is remote) or a focused one (if the visitor is a local). Journalists who work in local papers describe hyperlocalization as a way to take content that is broad (i.e., syndicated from a national/regional sources) and modifying it to boost local relevance.

DISCUSSION AND FUTURE WORK

Content personalization offers significant opportunities for content producers and readers alike. While PersaLog's implementation is designed to support the guidelines derived from our survey when possible, conventions and guidelines must be developed outside of the technical infrastructure. For example, content providers can hide the fact that personalization is happening by simply disabling our "PersaLog" buttons. Future work may include developing best-practices that can be more broadly applied (e.g., not inferring personalization features or accessing data that will not be used). Other unintended consequences of personalization (e.g., the diminishing value of comments when everyone sees different content) may lead to technical innovations that provide balanced personal–global spaces. These implications require serious consideration about the underlying values a site would like to engender.

In designing and experimenting with PersaLog, we identified specific and general personalization mechanisms. For example, some textual annotations are useful for drawing attention in a visualization. These could be packaged as visualization libraries to make them more viable for personalization. We have also begun to identify general design patterns for content personalization (e.g., personalized content should include "facts" that are a superset of the non-personalized). Deploying PersaLog in the wild, among practicing journalists, would allow us to test the ecological validity of our approach, iterate on the language and system design. With broader use and experimentation, re-usable design patterns will also likely emerge. Additional evaluations of the language (e.g., [28]) may also provide insights for future versions.

CONCLUSIONS

We introduced the PersaLog system and PERSALOG DSL. Like feed personalization, content personalization can enhance learning and behavioral changes in readers and increase engagement. In designing tools that support these features we have identified a number of unique properties and requirements that emerge in the context of news personalization. We report on a survey of journalism professionals that helped us define a set of guidelines for content personalization. Using this feedback, we have created a novel system that supports a broad set of personalizations that can be applied to existing and new (non-personalized) text and graphical content.

ACKNOWLEDGEMENTS

This was supported by the NSF under grant IIS-1421438. We are gratefully to our many respondents for their input.

REFERENCES

1. Gregor Aisch, Quoctrung Bui, Amanda Cox, and Kevin Quealy. 2016. Where the Poor Live Longer: How Your

Area Compares. *The New York Times*, April 11, Available: *http://nyti.ms/1SryMtT*. (2016).

2. Gregor Aisch, Eric Buth, Matthew Block, Amanda Cox, and Kevin Quealy. 2015. The Best and Worst Places to Grow Up: How Your Area Compares. *The New York Times*, May 4, 2015, Available: *http://nyti.ms/1KGrkJM*. (2015).
3. Jon Bentley. 1986. Programming Pearls: Little Languages. *Commun. ACM* 29, 8 (Aug. 1986), 711–721. DOI:http://dx.doi.org/10.1145/6424.315691
4. Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. 2005. Automation and Customization of Rendered Web Pages. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology (UIST '05)*. ACM, New York, NY, USA, 163–172. DOI: http://dx.doi.org/10.1145/1095034.1095062
5. Engin Bozdag. 2013. Bias in algorithmic filtering and personalization. *Ethics and Information Technology* 15, 3 (2013), 209–227.
6. Paul De Bra, Evgeny Knutov, David Smits, Natalia Stash, and Vinicius F. C. Ramos. 2013. GALE: a generic open source extensible adaptation engine. *New Review of Hypermedia and Multimedia* 19, 2 (2013), 182–212. DOI: http://dx.doi.org/10.1080/13614568.2013.806961
7. Paul De Bra, Natalia Stash, David Smits, Cristóbal Romero, and Sebastián Ventura. 2007. *Authoring and Management Tools for Adaptive Educational Hypermedia Systems: The AHA! Case Study*. Springer Berlin Heidelberg, Berlin, Heidelberg, 285–308. DOI: http://dx.doi.org/10.1007/978-3-540-71974-8_11
8. Peter Brusilovsky and Eva Millán. 2007. *User Models for Adaptive Hypermedia and Adaptive Educational Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 3–53. DOI:http://dx.doi.org/10.1007/978-3-540-72079-9_1
9. Darla Cameron and Denise Lu. 2016. Personal interview, July 12. (2016).
10. Gina Masullo Chen, T Makana Chock, Hillary Gozigian, Ryan Rogers, Arushi Sen, Valarie N Schweisberger, Joseph Steinhardt, and Yi Wang. 2011. Personalizing news websites attracts young readers. *Newspaper Research J.* 32, 4 (2011), 22–38.
11. Pascal R Chesnais, Matthew J Mucklo, Jonathan Sheena, and others. 1995. The Fishwrap personalized news system. In *2nd International Workshop Community Networking, 1995*. IEEE, New York, NY, 275–282.
12. Kevyn Collins-Thompson, Paul N. Bennett, Ryen W. White, Sebastian de la Chica, and David Sontag. 2011. Personalizing Web Search Results by Reading Level. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*. ACM, New York, NY, USA, 403–412. DOI: http://dx.doi.org/10.1145/2063576.2063639
13. Diana I Cordova and Mark R Lepper. 1996. Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *J. of Educational Psychology* 88, 4 (1996), 715.
14. Allen Cypher, Mira Dontcheva, Tessa Lau, and Jeffrey Nichols. 2010. *No Code Required: Giving Users Tools to Transform the Web*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
15. Nada Dabbagh and Anastasia Kitsantas. 2012. Personal Learning Environments, social media, and self-regulated learning: A natural formula for connecting formal and informal learning. *The Internet and higher education* 15, 1 (2012), 3–8.
16. Judy Davis-Dorsey, Steven M Ross, and Gary R Morrison. 1991. The role of rewording and context personalization in the solving of mathematical word problems. *J. of Educational Psychology* 83, 1 (1991), 61.
17. Chrysanne Di Marco, Peter Bray, H Dominic Covvey, Donald D Cowan, Vic Di Ciccio, Eduard Hovy, Joan Lipa, and Cathy Yang. 2006. Authoring and generation of individualized patient education materials. In *AMIA Annual Symposium Proceedings*, Vol. 2006. American Medical Informatics Association, Bethesda, MD, 195.
18. Paula J Durlach and Alan M Lesgold. 2012. *Adaptive technologies for training and education*. Cambridge University Press, New York, NY.
19. Facebook. 2017. Ads Guide. *https://www.facebook.com/business/ads-guide/*. (2017).
20. Leah Findlater and Joanna McGrenere. 2008. Impact of Screen Size on Performance, Awareness, and User Satisfaction with Adaptive Graphical User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1247–1256. DOI: http://dx.doi.org/10.1145/1357054.1357249
21. Jonathan G.K. Foss and Alexandra I. Cristea. 2010. The Next Generation Authoring Adaptive Hypermedia: Using and Evaluating the MOT3.0 and PEAL Tools. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (HT '10)*. ACM, New York, NY, USA, 83–92. DOI:http://dx.doi.org/10.1145/1810617.1810633
22. Evgeniy Gabrilovich, Susan Dumais, and Eric Horvitz. 2004. Newsjunkie: Providing Personalized Newsfeeds via Analysis of Information Novelty. In *Proceedings of the 13th International Conference on World Wide Web (WWW '04)*. ACM, New York, NY, USA, 482–490. DOI: http://dx.doi.org/10.1145/988672.988738
23. Krzysztof Z. Gajos, Mary Czerwinski, Desney S. Tan, and Daniel S. Weld. 2006. Exploring the Design Space for Adaptive Graphical User Interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '06)*. ACM, New York, NY, USA, 201–208. DOI: http://dx.doi.org/10.1145/1133265.1133306

24. Krzysztof Z. Gajos, Katherine Everitt, Desney S. Tan, Mary Czerwinski, and Daniel S. Weld. 2008. Predictability and Accuracy in Adaptive User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1271–1274. DOI: http://dx.doi.org/10.1145/1357054.1357252

25. Arline T Geronimus, John Bound, and Lisa J Neidert. 1996. On the validity of using census geocode characteristics to proxy individual socioeconomic characteristics. *J. Amer. Statist. Assoc.* 91, 434 (1996), 529–537.

26. Stefan Göbel, Sandro Hardy, Viktor Wendel, Florian Mehm, and Ralf Steinmetz. 2010. Serious Games for Health: Personalized Exergames. In *Proceedings of the 18th ACM International Conference on Multimedia (MM '10)*. ACM, New York, NY, USA, 1663–1666. DOI: http://dx.doi.org/10.1145/1873951.1874316

27. Google. 2017. Google Ads. *https://www.google.com/ads/*. (2017).

28. T.R.G. Green and M. Petre. 1996. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages & Computing* 7, 2 (1996), 131 – 174. DOI: http://dx.doi.org/10.1006/jvlc.1996.0009

29. Jonathan Harper and Maneesh Agrawala. 2014. Deconstructing and Restyling D3 Visualizations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 253–262. DOI: http://dx.doi.org/10.1145/2642918.2647411

30. Kashmir Hill. 2016. How an internet mapping glitch turned a random Kansas farm into a digital hell. *Fusion*, April 10, Available: *http://fusion.net/story/287592/internet-mapping-glitch-kansas-farm/*. (2016).

31. Graeme Hirst, Chrysanne DiMarco, Eduard Hovy, and Kimberley Parsons. 1997. *Authoring and Generating Health-Education Documents That Are Tailored to the Needs of the Individual Patient*. Springer Vienna, Vienna, 107–118. DOI: http://dx.doi.org/10.1007/978-3-7091-2670-7_14

32. Kristina Höök. 2000. Steps to take before intelligent user interfaces become real. *Interacting with computers* 12, 4 (2000), 409–426.

33. B. Huffaker, M. Fomenkov, and k. claffy. 2011. *Geocompare: a comparison of public and commercial geolocation databases - Technical Report* . Technical Report. Cooperative Association for Internet Data Analysis (CAIDA).

34. Chris J. Hughes, Mike Armstrong, Rhianne Jones, and Michael Crabb. 2015. Responsive Design for Personalised Subtitles. In *Proceedings of the 12th Web for All Conference (W4A '15)*. ACM, New York, NY, USA, Article 8, 4 pages. DOI: http://dx.doi.org/10.1145/2745555.2746650

35. Judy Kay. 2000. Stereotypes, Student Models and Scrutability. In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems (ITS '00)*. Springer-Verlag, London, UK, UK, 19–30. http://dl.acm.org/citation.cfm?id=648030.745813

36. Yea-Seul Kim, Jessica Hullman, and Maneesh Agrawala. 2016. Generating Personalized Spatial Analogies for Distances and Areas. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 38–48. DOI: http://dx.doi.org/10.1145/2858036.2858440

37. Aleksandra Klašnja-Milićević, Boban Vesin, Mirjana Ivanović, and Zoran Budimac. 2011. E-Learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education* 56, 3 (2011), 885–899.

38. Scott Klein. 2013. How To Edit 52,000 Stories at Once. *ProPublica Nerd Blog*, Jan. 24, 2013, Available: *https://goo.gl/5QipBA*. (2013).

39. Evgeny Knutov, Paul De Bra, and Mykola Pechenizkiy. 2009. AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. *New Review of Hypermedia and Multimedia* 15, 1 (2009), 5–38.

40. Andrew J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. 2011. The State of the Art in End-user Software Engineering. *ACM Comput. Surv.* 43, 3, Article 21 (April 2011), 44 pages. DOI: http://dx.doi.org/10.1145/1922649.1922658

41. Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: Automating & Sharing How-to Knowledge in the Enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1719–1728. DOI: http://dx.doi.org/10.1145/1357054.1357323

42. Yuping Liu and LJ Shrum. 2002. What is interactivity and is it always such a good thing? Implications of definition, person, and situation for the influence of interactivity on advertising effectiveness. *J. of Advertising* 31, 4 (2002), 53–64.

43. Catherine McLoughlin and Mark JW Lee. 2010. Personalised and self regulated learning in the Web 2.0 era: International exemplars of innovative pedagogy using social software. *Australasian Journal of Educational Technology* 26, 1 (2010), 28–43.

44. MediumEditor Open Source Team. 2016. MediumEditor. *https://yabwe.github.io/medium-editor/*. (2016).

45. Ted Mellnik, Darla Cameron, Denise Lu, Emily Badger, and Kat Downs. 2016. America's great housing divide: Are you a winner or loser? *Washington Post*, April 28, 2016, Available: *http://wapo.st/housing*. (2016).

46. Maria Milosavljevic. 1997. *Augmenting the User's Knowledge via Comparison*. Springer Vienna, Vienna, 119–130. DOI: http://dx.doi.org/10.1007/978-3-7091-2670-7_15

47. Alejandro Montes García, Paul De Bra, George H.L. Fletcher, and Mykola Pechenizkiy. 2014. A DSL Based on CSS for Hypertext Adaptation. In *Proceedings of the 25th ACM Conference on Hypertext and Social Media (HT '14)*. ACM, New York, NY, USA, 313–315. DOI: http://dx.doi.org/10.1145/2631775.2631782

48. Dan Murray and Kevan Durrell. 2000. *Inferring Demographic Attributes of Anonymous Internet Users*. Springer Berlin Heidelberg, Berlin, Heidelberg, 7–20. DOI:http://dx.doi.org/10.1007/3-540-44934-5_1

49. Reinhard Oppermann and R Rasher. 1997. Adaptability and adaptivity in learning systems. *Knowledge transfer* 2 (1997), 173–179.

50. Shahida M. Parvez and Glenn D. Blank. 2008. *Individualizing Tutoring with Learning Style Based Feedback*. Springer Berlin Heidelberg, Berlin, Heidelberg, 291–301. DOI: http://dx.doi.org/10.1007/978-3-540-69132-7_33

51. Paul A Pavlou and David W Stewart. 2000. Measuring the effects and effectiveness of interactive advertising: A research agenda. *J. of Interactive Advertising* 1, 1 (2000), 61–77.

52. Kevin Quealy. 2016. Personal interview, July 1. (2016).

53. Kevin Quealy and Margot Sanger-Katz. 2015. The Experts Were Wrong About the Best Places for Better and Cheaper Health Care. *The New York Times*, December 15, 2016, Available: *http://nyti.ms/1IRnP6J*. (2015).

54. Ehud Reiter, Robert Dale, and Zhiwei Feng. 2000. *Building natural language generation systems*. MIT Press, Cambridge, MA.

55. Mark Christopher Roberts. 2007. *Measuring the relationship between journalistic transparency and credibility*. Ph.D. Dissertation. University of South Carolina.

56. Celette Sugg Skinner, Victor J Strecher, and Harm Hospers. 1994. Physicians' recommendations for mammography: do tailored messages make a difference? *American Journal of Public Health* 84, 1 (1994), 43–49.

57. Michael Strickland, Archie Tse, Matthew Ericson, and Tom Giratikanon. 2016. ArchieML. *http://archieml.org/*. (2016).

58. Margaret Sullivan. 2015. When News Gets (Too?) Personal. *The New York Times*, Public Editor's Journal, Dec. 17, *http://nyti.ms/1T3H9hs*. (2015).

59. The New York Times. 2015. Innovation. May 24, 2015. (2015).

60. Theophanis Tsandilas and m. c. schraefel. 2005. An Empirical Assessment of Adaptation Techniques. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. ACM, New York, NY, USA, 2009–2012. DOI: http://dx.doi.org/10.1145/1056808.1057079

61. Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3227–3231. DOI: http://dx.doi.org/10.1145/2858036.2858556

62. Shan Wang. 2016. The Los Angeles Times built its own journalist-friendly story editor, and it's now rolling out to all of Tronc. *Nieman Lab*, August 10, 2016, Available: *goo.gl/c7wFVm*. (2016).

63. Jeffrey Wong and Jason I. Hong. 2007. Making Mashups with Marmite: Towards End-user Programming for the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 1435–1444. DOI: http://dx.doi.org/10.1145/1240624.1240842