Web-based Just-In-Time Retrieval for Cultural Content

Jörg Schlötterer
University of Passau
Innstrasse 33a
Passau, Germany
joerg.schloetterer@uni-passau.de

Christin Seifert
University of Passau
Innstrasse 33a
Passau, Germany
christin.seifert@uni-passau.de

Michael Granitzer
University of Passau
Innstrasse 33a
Passau, Germany
michael.granitzer@uni-passau.de

ABSTRACT

Digital content providers of cultural resources face the challenge of disseminating their content to interested users – either because users do not know the existence of resources or do not know the access points. We propose to apply just-in-time retrieval mechanisms to bring the content to the users in a web-based scenario. Our first prototype is a Chrome extension which proposes Europeana content based on the current user context, i.e., the current web site. We think that our approach is promising for scenarios where users are not aware of available content and therefore can and do not explicitly state an information need.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: [query formulation, relevance feedback]; H.3.7 [**Digital Libraries**]: [dissemination, user issues]

Keywords

Cultural Heritage, Personalized Search, Browser Extension, Just-in-Time-Retrieval

1. INTRODUCTION

The Web contains a plethora of valuable cultural, scientific and educational resources like scientific papers, tutorials and digitized museum artefacts – mostly hidden in the long-tail of the Web [1]. This means, that a regular web user does either not know the resources exist or cannot find it with a general search engine like Google because of the lower popularity of those sites reflected in the search ranking. Specialized search interfaces like Europeana[1] and Collections Trust[2] aim to bridge this content-user gap and provide access to long-tail contents. While this greatly improves the accessibility of the content, from the users' perspective it still requires the knowledge of the specialized search engine and an explicit formulation of a search query.

A recent initiative, so called edit-a-thons enhance the online encyclopedia Wikipedia with external resources, e.g. cultural content from digital libraries. Their goal is to bridge the content-user gap by bringing together Wikipedians with providers of long-tail contents, such as museums, libraries and archives and disseminating their contents via a major web hub (namely Wikipedia). Europeana, the European aggregator for digital, cultural heritage content, co-organized Wikipedia edit-a-thons in 2012 and 2013. In a case study they summarize the experiences in their 3 edit-a-thons with being successful by enhancing the Wikipedia and providing higher quality content to the public, content providers get a new distribution channel for their content [7].

Current work in personalized access to cultural heritage mainly focuses on server-side [4] or mobile solutions [10]. While the former requires additional effort from content-providers, which may be not feasible for small digital libraries, the latter applies to (mobile) application areas, differing from a typical web browsing setting. We investigate how access to cultural heritage can be realized in a typical web browsing setting without burdening content providers.

In information retrieval research, just-in-time information retrieval encompasses approaches to automatically present search results to users based on the users' context in a non-intrusive manner [9]. In just-in-time information retrieval, the user does not explicitly state her information need, i.e. does not formulate an explicit query. Studies showed that users access more information than with traditional search engines [9].

In previous work [3] we described two general usage scenarios for digital cultural heritage: content-consumption and content-creation. In the content-consumption scenario the user accesses the resources for personal use, whereas in the content-creation scenario the content is integrated or linked in a web page or online social media (blog, news article, online encyclopedia, tweet) and thus disseminated further. In this paper we report work in progress to just-in-time retrieval of cultural content in a web-based setting for the content-consumption scenario. We propose to use a web browser extension which observes the current context of the user (the page) and presents relevant resources – if available.

First, we describe the usage scenario in detail in section 2. The general approach and design decision to solve this scenario are presented in section 3, followed by our prototypical implementation in section 4. We conclude and provide an outlook on future directions in section 5.

[1] `http://europeana.eu`
[2] `http://www.collectionstrust.org.uk/`

PATCH '14 Haifa, Israel

2. DETAILED SCENARIO

We follow the Scenario-Persona approach, where a Scenario is defined as "a concise description of a persona using a software-based product to achieve a goal" [2]. Our persona is represented by Bob K., a digital native, experienced in using web-based tools and not afraid of using new technologies. Bob has no experience on using the facilities of cultural heritage providers and in general only limited knowledge in the cultural heritage domain.

To acquire information on a certain topic, he will probably start with Google or Wikipedia and has no knowledge about specialized search interfaces like Europeana or web sites of memory organizations in his area. We assume that either way he will end up on Wikipedia to do his inquiry. Most likely, the resources found on Wikipedia might give him a direction of how to satisfy his information need, providing basic information and maybe some historic background-information. But it will not suffice to get a complete picture, he will need more detailed information.

The just-in-time retrieval process for cultural content will utilize the results provided by Wikipedia and maybe also the initial query to form further queries, sending the queries to different content providers and aggregating their results. It furthermore has information on Bob's location via the browser's Geolocation API [8]. The returned resources will be presented to Bob in an appropriate way, according to the nature of the underlying data. These presentations can range from simple textual result lists to graphical visualizations of the results. e.g., a time line.

The search system also returns links to institutions (i.e. museums) that provide content and/or documents on the subject (e.g. photos) or maybe even hold a relevant object at their exhibition. As a result, Bob will not only have information on the subject in general, but he will also be provided with information on (i) specific objects that are related to the subject and held by local museums, (ii) where to look at/use those objects in a museum nearby, (iii) where to get additional, first-hand information on those objects.

While we described a very specific scenario here, in general the retrieval process is not limited to Wikipedia, but will be triggered on every web page for which additional information is desirable and provide the desired information.

3. APPROACH

To solve the scenario described above, i.e., to enrich the user's browsing experience with cultural content, additional functionality needs to be added to existing web pages, which is called "JavaScript injection". By executing injected JavaScript, the look and feel of web pages can be altered and additional information and functionality can be added. This injection can be achieved by (i) a bookmarklet, executing JavaScript commands stored as bookmark, (ii) a browser extension - once installed, extending the browser's functionality on all web sites, or (iii) a widget, embedded into a web page server-side.

Although the initial burden to install a browser extension is a little higher compared to the other approaches (while the bookmarklet is simply stored as bookmark, the widget requires no user action at all), we chose to implement a browser extension because of the following advantages: The widget is limited to the web pages implementing it and the bookmarklet needs to be triggered on every page on which additional information is desirable, whereas the browser extension is applicable to all web pages without additional effort. Also, both, bookmarklet and widget are limited to the context of the current page and cannot account for additional information sources (e.g. browsing history), to tailor the injected cultural contents towards the user's interests.

After examining the extension possibilities of the major web browsers and their market share, we decided to start with an extension for Google Chrome for the following reasons: According to StatCounter[3] Google Chrome has the highest (and still growing) market share of 40%. Moreover, Firefox, Safari and Opera share similar extension architectures with Chrome, all based on standard web technologies, such as HTML, JavaScript and CSS. Thus, the results from a Google Chrome extension can easily be transferred to the aforementioned other browsers. Developing an extension for Google Chrome and porting the result to the browsers with a similar extension architecture will cover around 70% of the browsers in use.

4. IMPLEMENTATION

Our first prototype of an extension for the Google Chrome browser[4] provides personalized search results from the cultural heritage domain, based on the contents of the current web page or the contents of a selected paragraph within this web page. As back end for the automatically generated queries, we use the Europeana API, which can be easily exchanged by a more sophisticated personalized search system (aggregating information of different content providers and incorporating usage feedback) later on.

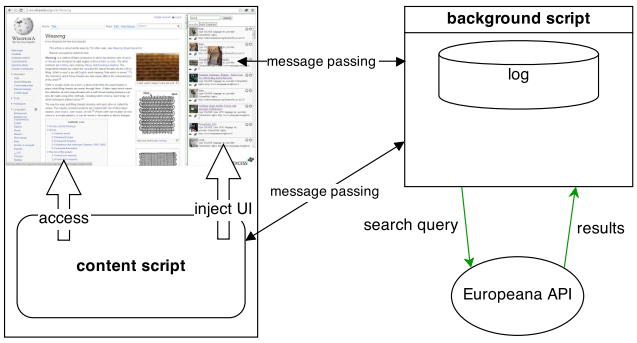

Figure 1: Architecture of the Google Chrome extension prototype

4.1 Architecture

Figure 1 shows an overview of the architecture. The basic architecture is already predetermined by the extension API of Chrome, partitioning the extension in (i) a *manifest file*, containing information about the extension and configuration settings such as permissions, (ii) *content scripts*, injected into web pages and having access to the particular DOM-tree and (iii) a *background page or script*, which is a single (long-running) instance of the extensions logic with access to the browser API.

The content script is responsible for retrieving and preprocessing the contents of the current page and forwarding

[3] http://gs.statcounter.com/

[4] http://mics.fim.uni-passau.de/demos-downloads/

the results to the background script, as well as for injecting the user interface into the page. The background script creates a query, based on the contents received from the content script, sends this query to the search back end and forwards the results to the injected user interface. The background script furthermore stores the user's interaction and feedback in a log.

4.2 User Interface

Figure 2 shows the basic user interface, injected in every web page, simulating the behaviour of a sidebar. The user can toggle the visibility of the interface globally for all pages with the EEXCESS icon 01.The search terms which triggered the presented results are displayed in the search field 02 and can be edited by the user. Adjustments to the query are logged, in order to be able to learn the user's query preferences. Each result item 03 is presented with a preview image 04, the title 05 and facets, such as type, language, provider and rights (if available) 06, along with further interaction possibilities: By clicking either the title or the preview image, an overlay is shown with an HTML representation of the resource, containing additional information. Presenting this information inside the current web page, enables us to keep track of dwell time on a particular result directly within the extension and does not draw the user's attention away from his initial intention.

Beneath storing potential views of a result item as implicit feedback, the user can give explicit feedback, by rating the result up or down with the buttons at 07. These ratings are stored in the Open Annotation format [11] and thus can be easily shared across different platforms. The link button 08 provides a reference URL of the resource for bookmarking or sharing it in an online social network.

The "options" tab 06 is a shortcut to the extension's options page, which is accessible via the extension administration page in Chrome as well. At the current state, it solely provides the ability to enter and edit demographic information.

4.3 Content Script

Separate instances of the content script are injected into every web page. Since the content script has full access to the DOM-tree of the particular page, it is responsible for mining the contents of the page and forwarding them to the background script. In the first approach, the three most frequent words of the current page or selected paragraph (if any) are used as query terms to retrieve recommendations.

Beneath mining the contents of the given web page, the content script also monitors the user's interactions with the page, such as mouse clicks or textual input. These interactions can serve as additional information for deducing the query terms. They allow us for example to establish search histories from search engines the user habitually uses. While search histories provide a great information source for personalization strategies [12], we do not expect users to extensively utilize the search interface in our injected sidebar (and do not intend having users formulate a massive amount of queries, but instead providing interesting recommendations automatically). Thus, by keeping track of the user's inputs, we have search histories available without any additional effort of the user.

The third task of the content script is the injection of the user interface (described in section 4.2) into the current web page via an *iframe*-element. The use of an iframe prevents the interface from directly interacting with the current page, due to the cross-origin policy, but is necessary to avoid inheriting CSS-styles of the current page. An iframe-element is advantageous in a further aspect as well: it allows the injection of other user interfaces, which are totally decoupled from the basic one. Since the injected user interfaces communicate directly with the background script via message passing, additional interfaces only need to implement the interface to the background script and no other component.

4.4 Background Script

Only a single instance of the background script exists for the whole extension. This script is responsible for the communication with the search back end and serves as a mediator between the user interface injected in a web page and the respective content script, since these two cannot communicate directly. User related information, retrieved from a content script (such as interactions on a web page or aggregated contents of this page) gets logged by the background script. Based on these logs, we plan to establish user profiles, in order to improve search result personalization. The logs contain also usage feedback about recommended resources, such as the viewing time of a recommended result or a rating for it.

To overcome limitations of the browsing history in the browser's API, the background script features an own history implementation, containing not only timestamps for the beginning of a visit on a certain web page, but instead storing the active dwell time on that particular page. "Active" in this case means, the browser and the tab with the particular page within the browser has the focus (switching to another application means switching the focus and thus, the visit ends). In addition, the referring URL (if any), important for deducing navigational paths or patterns, is stored explicitly along with the visit, while retrieving it via the Chrome API is quite cumbersome.

4.5 Data Storage

At the current point of time, all data handled within the prototype is stored locally on the client, since this is advantageous in terms of privacy concerns: the user has full control over the data stored and all the information resides within the client, not being transferred to any outside system. The drawback of this approach is that client-side stored logs are prone to be deleted by accident: when the user deletes her private data via the browser integrated function, all of the extension's data are swept away as well.

Basically, two mechanisms exist for storing data on the client: via Web Storage[5], depositing the data as stringified key-value pairs, or in an Indexed Database[6], which consists of object stores, holding records of key-value pairs. We decided to go with the latter, as it provides some significant advantages over Web Storage: In terms of the keys, additional data types, such as Numbers, Date- and Array-objects are allowed. Also, duplicate values for keys can be stored (and iterated). The values to be stored must be supported by the structured clone algorithm, providing some benefits over JSON-serialization, such as being able to handle Blob-, File- and FileList-objects for example. The main advantage of the Indexed Database over Web Storage is the efficient retrieval of records via indexes.

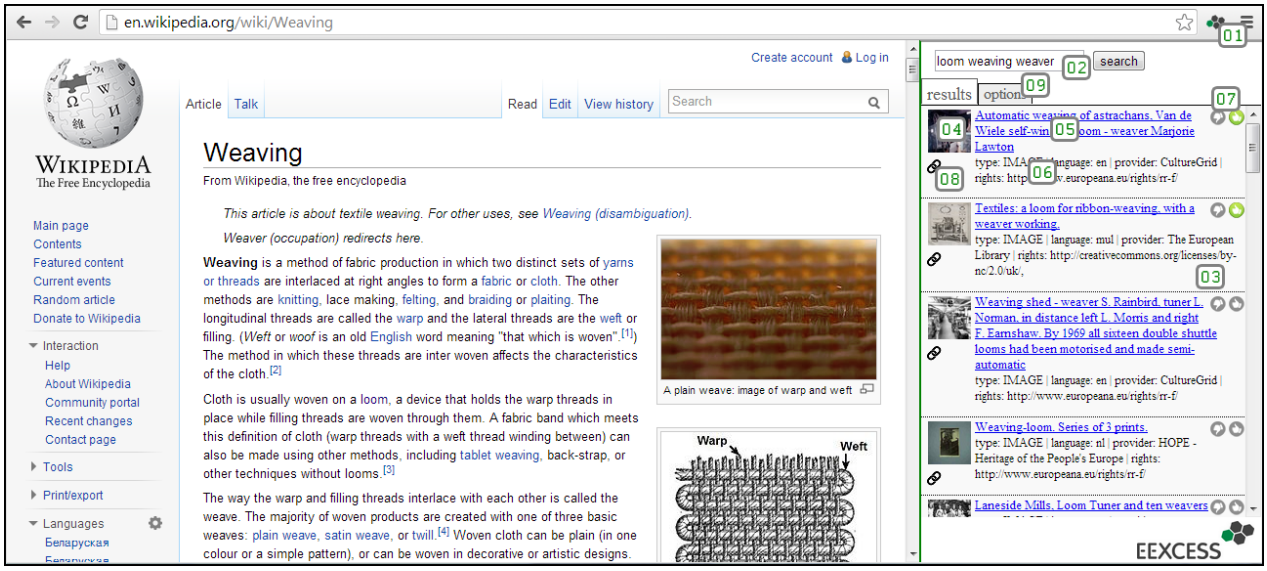

Figure 2: Screenshot of injected user interface

5. CONCLUSIONS AND FUTURE WORK

We described challenges and opportunities for enriching a user's browsing experience with cultural contents in a web setting and presented our approach to it: a first prototype of a Google Chrome extension for just-in-time retrieval. In the future we will include more sophisticated context detection and query (re-)formulation methods. We will establish user profiles, capturing a user's interests and identify the need for additional information by means of task detection. Further, we will research alternative result presentation methods, e.g., using information visualizations providing an overview of the retrieval results. In addition, we aim to develop a standardized data model and API for connecting additional data providers. We plan to evaluate the usability of the interface with a group of end users and the quality of recommendations with domain experts.

Following the current debate on privacy aspects of user data we will provide means for users to adapt their privacy settings to their personal need within the extension. Further, we will research privacy-preserving personalization approaches, e.g., by not personalizing for single users but for user groups and thus guarantee k-anonymity.

6. ACKNOWLEDGEMENTS

The presented work was developed within the EEXCESS project funded by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement number 600601.

7. REFERENCES

[1] A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1–4):69 – 77, 2000.

[2] A. Cooper. *The inmates are running the asylum: Why high tech products drive us crazy and how to restore the sanity*. Pearson Higher Education, 2004.

[3] M. Granitzer, C. Seifert, S. Russegger, and K. Tochtermann. Unfolding cultural, educational and scientific long-tail content in the web. In S. Berkovsky, E. Herder, P. Lops, and O. C. Santos, editors, *UMAP Extended Proceedings*, volume 997, 6 2013.

[4] C. Hampson, E. Bailey, G. Munnelly, S. Lawless, and O. Conlan. Dynamic personalisation for digital cultural heritage collections. In *UMAP Workshops – Patch 2013*, 2013.

[5] I. Hickson. Web Storage. W3C Recommendation, 2013. `http://www.w3.org/TR/webstorage/`.

[6] N. Mehta, J. Sicking, E. Graff, A. Popescu, J. Orlow, and J. Bell. Indexed Database API. W3C Candidate Recommendation, 2013. `http://www.w3.org/TR/IndexedDB/`.

[7] G. Oskam, J. Andersson, and Álex Hinojo. Case study: Europeana edit-a-thon. online, 2013.

[8] A. Popescu. Geolocation API Specification. W3C Recommendation, 2013. `http://www.w3.org/TR/geolocation-API/`.

[9] B. J. Rhodes. *Just-In-Time Information Retrieval*. PhD thesis, Massachusetts Institute of Technology, 2000.

[10] T. Ruotsalo, K. Haav, A. Stoyanov, S. Roche, E. Fani, R. Deliai, E. Mäkelä, T. Kauppinen, and E. Hyvönen. Smartmuseum: A mobile recommender system for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 20(0):50 – 67, 2013.

[11] R. Sanderson, P. Ciccarese, and H. V. de Sompel. Open Annotation Data Model. W3C Community Draft, 2013. `http://www.openannotation.org/spec/core/`.

[12] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 718–723, New York, NY, USA, 2006. ACM.