WASP: Web Archiving and Search Personalized

Johannes Kiesel
Bauhaus-Universität Weimar
Weimar, Germany
johannes.kiesel@uni-weimar.de

Arjen P. de Vries
Radboud University
Nijmegen, The Netherlands
a.devries@cs.ru.nl

Matthias Hagen
Martin-Luther-Universität
Halle-Wittenberg
Halle, Germany
matthias.hagen@informatik.
uni-halle.de

Benno Stein
Bauhaus-Universität Weimar
Weimar, Germany
benno.stein@uni-weimar.de

Martin Potthast
Leipzig University
Leipzig, Germany
martin.potthast@uni-leipzig.de

ABSTRACT

Logging and re-finding the information we encounter every day while browsing the web is a non-trivial task that is, at best, inadequately supported by existing tools. It is time to take another step forward: we introduce WASP, a fully functional prototype of a personal web archive and search system, which is available open source and as an executable Docker image. Based on the experiences and insights gained while designing and using WASP, we outline how personal web archive and search systems can be implemented, discuss what technological and privacy-related challenges such systems face, and propose a setup to evaluate their effectiveness. As a key insight, we argue that the indexing and retrieval for a personal archive search can be strongly tailored towards a specific user and their behavior on the visited pages compared to regular web search.

1 INTRODUCTION

Lifelogging[1] has become a common practice, as a result of the omnipresence of smartphones, smart watches and fitness trackers, and emerging technologies such as smart glasses, wearable technologies and sensor-enabled smart homes. Isn't it surprising that keeping track of one's online activities is comparably underdeveloped? Significant amount of work has been invested into understanding personal information management [10] and developing tools to support it, including the winner of the SIGIR 2014 Test of Time Award "Stuff I've Seen" (SIS) by Dumais et al. [6]. With a bit of irony however, neither SIS nor follow-up Phlat [5] are available today, even if the key insights gained have likely informed the development of Windows desktop search and intelligent assistant Cortana. Likewise, Spotlight on MacOS supports search over local documents and other digital assets. Both are integrated with the web browsers from Microsoft and Apple, respectively, to index browsing history. Meanwhile, the history tabs of modern Web browsers provide access to the history of the currently open browser as well as pages recently visited on other devices. However, current browsers do not align and integrate the browsing histories across devices, nor, apparently, do the aforementioned tools index the content of web pages visited, but only their titles and URLs. In fact, the possibility to track (let alone search) one's browsing history using off-the-shelf tools is still fairly limited.

In this context, it is not surprising that personal information access was one of the major topics discussed at the Third Strategic Workshop on Information Retrieval in Lorne (SWIRL 2018) [4]. The attendees noted that this problem, open for so long, has not been addressed adequately, and, worse, that it is an ever more daunting challenge to help people re-find and re-visit their online information and prior information interactions with these sources; as this information today resides in multiple devices and a large variety of information services, that each construct their own data silos and search APIs (if such access is offered at all). Specifically, the report mentions the high cost of entry for scientists as a major obstacle, where "there is substantial engineering required for a minimal working system: to fetch data from different silos, parse different data formats, and monitor user activity."

We propose to take a pragmatic "shortcut" and to establish empirically how far that workaround can bring us. Increasingly, access to our digital information takes place through the web browser as *the* interface. Therefore, we set out to develop WASP, a prototype system for *personal* web archiving and search. WASP saves one's personal web browsing history using state-of-the-art web archiving technology and offers a powerful retrieval interface over that history. This browser-focused setup enables the user to recall information they personally gathered without the need to deal with the large variety of information sources. Even if we do not cover the full range of digital objects that may accrue on a person's desktop and mobile devices, high-quality archival of web pages visited may capture a large fraction of the information we interact with.

In addition to a detailed technical description of WASP in Section 2, this paper reports on the observations that we made (Section 3) and the challenges for personal web archiving and search that we identified (Section 4) through our extensive use of the WASP prototype—which we provide both open source and as an executable Docker container so that others can use it within their research or personal lifelogging setup.[2,3]

[1] https://en.wikipedia.org/wiki/Lifelog

DESIRES 2018, August 2018, Bertinoro, Italy

[2] https://hub.docker.com/r/webis/wasp/
[3] https://github.com/webis-de/wasp

2 THE WASP PROTOTYPE

The *WASP*[4] prototype integrates existing archiving, indexing, and reproduction technology for the first time into a single application. Figure 1 illustrates how the user's browser interacts through WASP with the World Wide Web under the three usage scenarios archival, search, and reproduction of web pages, as detailed below.

2.1 Archiving Proxy and Indexing

After starting WASP, the user has to reconfigure his or her browser to accept WASP as forward proxy and to trust its certificate. WASP then archives all HTTP(S) requests and responses from and to the browser in the standard Web archiving format (WARC) (Figure 1 (a)). This is achieved using the Internet Archive's `warcprox` software,[5] whose WARC contain all the information necessary to reproduce an archived browsing session at a later time.

In order to enable searching the archived content, we devised a software component that monitors WARC files and automatically indexes HTML responses and their corresponding requests in an ElasticSearch index.[6] In detail, we use the Lemur project's WARC parser[7] and Apache's `HttpClient` library[8] to read HTTP messages as they are appended to the WARC files. The title and text of the HTTP responses that have the MIME type HTML are extracted from responses using the Jericho HTML Parser library.[9] The title and text of the HTTP response is indexed along with the corresponding HTTP request's time and URL. Later page revisits (identified by `warcprox` through hash value matching on HTTP responses) are added to the response's record in the index. When the index is queried, the aggregation of requests avoids duplicate results in case a page is visited more than once.

Even if web pages vanish or change, WASP can reproduce the content the user saw in the past using the Web archiving toolkit `pywb`.[10] Like our automatic indexing setup described above, `pywb` monitors and indexes changes to the web archives. While the ElasticSearch index is tailored toward search within the HTML content of the archived web pages, the `pywb` index is tailored towards retrieving the HTTP response corresponding to a given HTTP request, enabling efficient reproduction of pages from the personal archive.

2.2 Search Interface

Access to the archived web pages is provided using the ElasticSearch index detailed in Section 2.1 (Figure 1 (b)). Under a configurable port, WASP provides the user with a basic search engine. Figure 2 shows a screenshot of the interface. Unlike regular web search engines, WASP's interface provides controls to specify the time the user recall visiting the desired web page ①, ②, ③[11] in addition to the familiar query box ④. Web pages are retrieved by matching query words against the title and contents of web pages visited in the specified time interval. ElasticSearch's highlight feature is used to generate query-related snippets for the results ⑨.

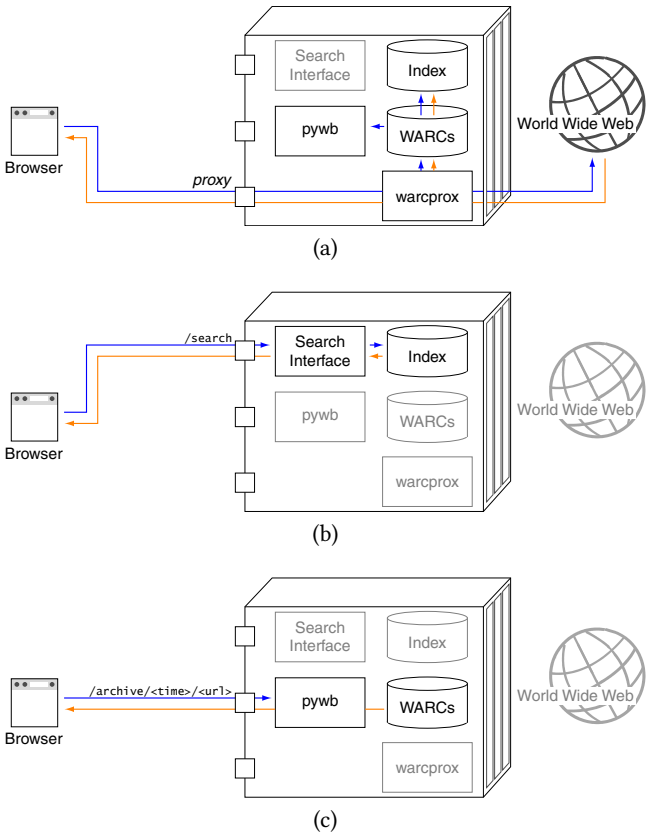

Figure 1: Architecture of the prototype: (a) during regular browsing, the container works as a forward proxy that stores all requests (►) and responses (◄) in web archive files (WARCs) and indexes them; (b) when browsing to localhost:<search-port>/search, the browser shows our search interface (Figure 2), where results link to (c) the reproduction server, which serves content from the WARC that (fuzzy) matches a specific time and URL.

A difference to a regular search engine results page is that in WASP, each result item consists of two hyperlinks: one resolving the URL to the live web ⑥ as usual, and another one pointing to the archived version of the web page. This latter hyperlink refers to the port of the WASP container's reproduction proxy and the access time and URL of the web page that should be reproduced. In case several non-identical versions of the same page are found in the requested interval, the prototype displays all of them as separate results. However, we expect that more mature personal web archiving and search systems will rather condense the different versions of a web page, especially when the context of the query terms is similar in the versions. The resulting user experience offers key advantages with respect to search users' privacy: search activities remain local to WASP, and the user is left in control whether to visit the live web page (without leaking their preferences to another search engine), or to be satisfied with the archived result.

[4] WASP is short for Web Archiving and Search, Personalized
[5] https://github.com/internetarchive/warcprox
[6] https://www.elastic.co/
[7] http://www.lemurproject.org/clueweb09/workingWithWARCFiles.php
[8] https://hc.apache.org/httpcomponents-client-ga/
[9] http://jericho.htmlparser.net/docs/index.html
[10] https://github.com/webrecorder/pywb
[11] Date and time picker widget: https://eonasdan.github.io/bootstrap-datetimepicker/

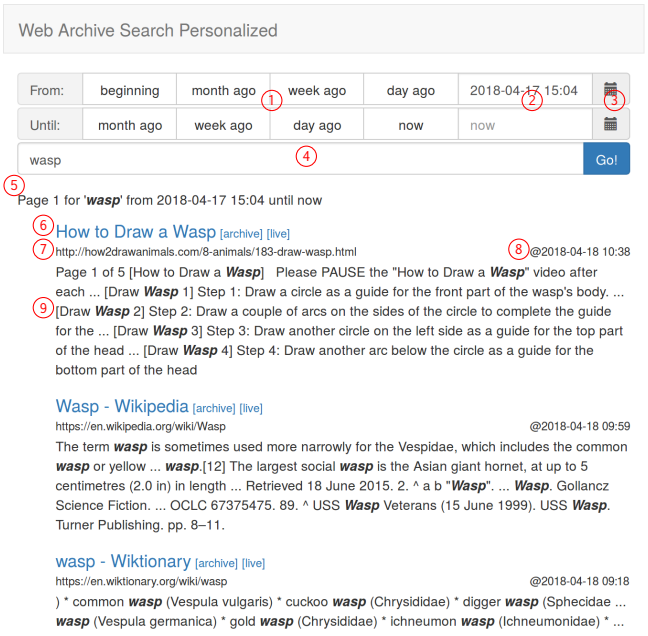

Figure 2: Search interface for WASP: ① shortcuts for frequently used time settings; ② selected query time interval; ③ date and time picker for exact time specification; ④ query box; ⑤ description of current result page; ⑥ title of result with links to archived and live version; ⑦ URL of the result; ⑧ archive time of the result; ⑨ snippet for the result.

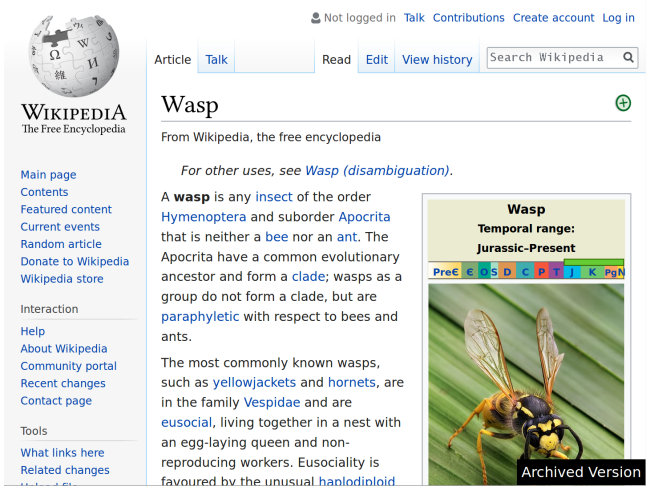

Figure 3: Screenshot of a web page reproduced from the archive. pywb is configured to insert a small black banner at the bottom right of the browser viewport to remind users that they are viewing an archived page.

2.3 Reproduction Server

When using a personal Web archive in a re-finding scenario, WASP fulfills the need of users to access information from their browsing history using pywb; a state-of-the-art web page reproduction software which uses intricate URL rewriting and code injection to serve the archived web pages like they were originally received (Figure 1 (c)). Through the use of specific URLs, pywb can serve multiple versions of the same web page. WASP's search interface uses this feature to refer the user to exactly that version of the web page that corresponds to the clicked result link. In order to avoid confusion on the user's side as to whether or not they are browsing within the archive, a small black banner is inserted and fixed to the bottom right corner of the browser viewport for all pages that are reproduced from the archive (cf. Figure 3).

3 QUALITATIVE EVALUATION

Given that the WASP prototype became operational only recently, the ongoing evaluation of its archiving and retrieval quality is still in its infancy. Nevertheless, since we have been using the prototype, this section reports on insights gathered so far, namely the results of an error analysis regarding archiving quality, and an outline of evaluation methodology regarding retrieval quality.

3.1 Archiving Quality: Error Analysis

When revisiting an archived web page, one naturally expects the version reproduced from the archive to look and behave exactly the same as the live version did at the time of archiving. Yet, technological difficulties may prevent the faithful reproduction of an archived web page. Since it is usually impractical for WASP to take web server snapshots, WASP will only capture a page's client side. Therefore, only a subset of the potential server interactions end up being represented in the archive and available for the reproduction: the scrolling, clicking, form submissions, video and audio stream playback, etc. that the user performed on the live web page. If user interactions on the archived web page trigger unseen requests to the web server, reproducing the archived web page will either do nothing, show an error, or stop working.

However, even in the case the user repeats the same basic interactions on the archived page that they performed on the live page, only about half of web pages can be reproduced flawlessly [9]. These reproduction errors mostly stem from randomized requests. Indeed, in about two-third of flawed reproductions, the errors are on the level of missing advertisements or similar. While pywb replaces the JavaScript random number generator by a deterministic one, this only affects the archived page and does not fully solve the problem: different timings in the network communications lead to a varying execution order and thus a different order of pop-requests from the "random" number sequence. To greater effect, pywb employs a fuzzy matching of GET parameters that ignores some of the parameters that it assumes to have random values (e.g., session ids), be it by the parameter name or by a hash-like appearance of the parameter value. While it is unclear how many false positives this process introduces, it naturally can't find all random parameters as there exists no standard whatsoever in this regard.

Another interesting problem for web archiving we noticed are push notifications: while they are properly recorded, it remains a difficult choice if and when to trigger them during the reproduction of a web page. Should the trigger time be based on the time spent on the page or based on other events?

Finally, we found that differences between browsers can also affect the reproduction quality. Though this had only minor effects on our experience with WASP so far, the ongoing development of the web technology stack may render old web pages in the archive less reproducible in the long run. For an example, consider the ongoing demise of Flash as a major container for dynamic content. In this regard, old versions of browsers and even old versions of operating systems may need to be kept, which is a definite requirement for web archiving in general, and also possible based on WASP's use of Docker containers, though not necessarily important for our usage scenario of personal web archiving.

3.2 Retrieval Quality Evaluation: An Outline

In principle, it should be easier to re-find something in a personal web archive than using some commercial search engine on the live web. Since a personal archive will typically be many orders of magnitude smaller, not as many candidate results for simple queries exist as on the live web. Ideally, compared to finding a needle in the huge haystack of the web, with a tailored search interface for one's smaller personal archive, the ratio of needles to hay is much higher in a re-finding scenario than in general web search. Still, since WASP is a prototype that was created very recently, we can only provide anecdotes of retrieval problems and sketch how we want to evaluate whether WASP actually helps to re-find needles.

The main evaluation scenarios we envision is re-finding something a user recalls having seen earlier on the web. Such re-finding intents will be different from the frequent re-visit patterns users show on the web [1] since their purpose is not to visit some favorite page but to check some information seen before. In this regard, we do not believe that, at the time of visiting a web page the first time around, users will have enough foresight and presence of mind to anticipate its future uses and hence place a bookmark, rendering a search in their personal archive indispensable.

We used WASP for one week in an informal self-experiment to figure out what problems arise and what should thus be integrated in a formal evaluation. The most obvious problem that differs from the general web search scenario is that of dealing with several versions of the same web page. During our short-term usage of WASP, we found that most retrieved web pages are actually relevant, but that the result lists are cluttered with different versions of the same web page that were—with respect to our information needs—practically identical; as predicted by a recent retrievability study of Web archive search [12]. A probably even more difficult problem, but one that our scenario shares with general web search, arises from the fact that nowadays web pages request a large part of their content dynamically and only if necessary. A good example of this is the Twitter timeline: while scrolling through the timeline, more tweets are requested from the server. Since WASP is currently limited to indexing HTML responses, it catches only some parts of the tweets (see Figure 4), which turn out to be HTML templates requested via Ajax for integration into the Twitter page.

Based on these observations, we propose the following evaluation setup for personal web archives. Since re-finding in personal web archives has not been part of any evaluation campaign so far, a respective set of topics and user interactions has to be built up

Figure 4: Example of dynamic HTML content in WASP: (a) original tweet as it appeared while scrolling down the Twitter timeline (b) Twitter card as it was requested for display, archived, and indexed.

front. Besides monitoring user queries against WASP's search functionality for users who agree to share parts of their browsing and search activity, one will periodically trigger active users of WASP with a re-finding game similar to PageHunt [11]. The user will be shown the screenshot of a page they have seen, or only parts thereof (e.g., only the color scheme of the layout), or will be asked to re-find a piece of information they have seen a given period of time ago (e.g., three days ago, two weeks ago, etc.). Their task will be to come up with a sequence of queries (and clicks) such that in the end the prescribed web page appears in the top-k ranks of WASP's retrieval component. In such cases, the desired item will be known for evaluation purposes and the re-finding task can have several difficulty levels (showing full information vs. only color scheme, target information at top of a page or only requested upon interaction, etc.). To measure retrieval success, the length of real and the comparably artificial re-finding query and click sequences can be measured as well as the specificity of the queries contrasted by the size of the personal collection. But of course, the overall interesting measure will be for how many real re-finding tasks the users are able to pull out the desired result from their personal archive—their needle stack.

4 DISCUSSION AND LESSONS LEARNED

Our primary goal with WASP was to develop a vertical prototype of a web archiving and retrieval framework, which archives every web page and every request made by a web page, and then indexes everything archived. Based on first practical experiences with using WASP for our own respective web traffic, however, there are still

many things to be sorted out before we can claim a flawless retrieval experience. Unsurprisingly, the devil is in the details, but somewhat surprisingly, we will be forced to revisit the basic notions of what is a web page, what needs to be archived, and what needs to be indexed. This section discusses lessons learned, outlining a number of exciting future directions for research and development on web archiving and retrieval in general, and for WASP in particular.

4.1 Which pages to archive?

Although WASP currently follows "archive first, ask questions later," users of a personal archiving system likely do not wish for all their traffic to be archived, even if stored within their personal data space. Without specific measures, sensitive data will end up in the archive, e.g., banking pages, health-related browsing, as well as browsing sessions with privacy-mode enabled (where users expect all traces of their activities to be purged after the browser is closed); users may not expect for such data to emerge in search results, weeks, months, or even years later. Furthermore, just as some users regularly clean or clear their browsing history, they will wish to clean or clear their archive. Similarly, it will be necessary to protect the personal archive from unauthorized access, analyze known and new attack vectors on the archiving setup, and formalize the security implications that stem from the use of such a system.

Based on these deliberations, it is clear that the user must be given fine-grained control over what sites or pages are archived, allowing for personal adjustments and policies. The recorded archive needs to be browseable, so that individual entries can be selected for removal. For more convenient browsing (both for cleaning and general re-finding), we suggest a screenshot-based interface as shown in Figure 5. At present, users can already influence which pages should not be archived using proxy-switching plugins available for all modern browsers that seamlessly integrate with WASP's proxy-based architecture (e.g., cf. Figure 6). Of course, specifying wildcard expressions hardly qualifies as a user-friendly interface for non-computer scientists, so that a better interface will be required in practice (e.g., using classification techniques similar to [7]).

Under some circumstances personal archiving systems could act on their own behalf to allow for an improved experience of the archived page, by archiving content the users did not request themselves. This possibility leads to several new research questions. For example, should all videos on a visited page be requested and archived, so that the user can watch them later on from their archive? Or in general, should the system predict and simulate interactions that the user may later want to do on the archived page to archive the corresponding resources while they are still available? Moreover, should the system perform such a simulation multiple times in order to detect the randomness in the web page's requests and consider this information in the reproduction?

4.2 Which pages to index?

While a comprehensive archive is necessary for a high-quality reproduction of web pages, not everything that the browser receives is actually of interest to the user. From our own web browsing habits, we can informally tell that many pages opened are not relevant for future retrieval, because they are dismissed upon first glance (e.g., pop-ups) or not even looked-at at all.

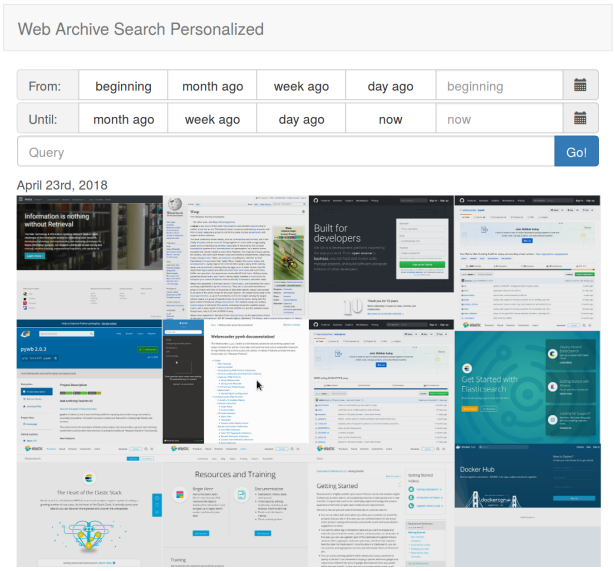

Figure 5: Screenshot mode mockup; the screenshot in the 2nd row and 2nd column is highlighted by mouse-over.

Figure 6: Firefox toolbar indicating archiving is activated. The context-menu of this icon allows to turn of the proxy-usage. thereby implementing a "pause-archiving" button.

Besides accidental page visits, another example of irrelevant pages may be found in more complex web applications. Take web-based RSS feed readers the likes of Feedly as an example: there is no need to index every page and every state of every page of the feed reader. Rather, the feed items to which the user pays attention are of interest for indexing, since only they are the ones the user may eventually remember and wish to revisit. In this regard, two cases can be distinguished, namely the case where feed items are displayed only partially, so that the user has to click on a link pointing to an external web page to consume a piece of content, and the case where feed items are displayed in full on the feed reader's page. The former case is straightforward, since a click indicates user attention, so that the feed reader's page can be entirely ignored. In the latter case, however, every feed item the user reads should be indexed, whereas the ones the user skips should not so as not to pollute the user's personal search results.

More generally, all kinds of portal pages and doorway pages, ranging from newspaper front pages via social network pages to search results pages are candidates for omission. Analyzing the user's browsing behavior gives evidence which page they sufficiently scrutinized for it to be indexed. If a user spends time reading the headlines and excerpts of a front page, this would suggest to index that page, but may be difficult to discern in practice. Otherwise, a user's behavior may be used as implicit relevance feedback to be incorporated into tailored retrieval models.

4.3 What is the document unit for indexing?

In its present form, WASP archives everything that runs under a given URL—including GET parameters, but excluding fragment identifiers—as one unit. Just like in regular search, not every piece of content is relevant for indexing. Main content extraction is an obvious solution to this problem, but the current state-of-the-art frequently fails on pages where many small pieces of content can be found. Furthermore, many websites today spread one coherent piece of content over many sub-pages (so-called pagination). For instance, news publishers often employ pagination, forcing readers to switch pages (possibly to serve extra display ads or improve engagement metrics that determine the value of the display ads shown on the publisher's site). For archive retrieval purposes, however, pagination can be detrimental, penalizing the relevance of a paginated news article to a query, since only parts of the article are scored at a time.

On the other hand, physical pages are also not necessarily atomic: many web pages built with modern web design tools are single-page applications, where different pieces of content are shown upon user request under the same URL. For instance, a blog platform may show each blog post requested by a user simply by loading it in the background using a JavaScript-based AJAX request, and replacing the currently shown post with a new one. In this case, the perfect web archive search would identify the single posts and index them separately, injecting code upon reproduction that replaces the displayed post with the desired one. Currently, we are technologically far from such a feature. In a different case, like the Twitter timeline, a web page consists of several (possibly independent) content segments. Again, each such segment should be indexed separately for an appropriate relevance computation. To meet this challenge, web pages should be segmented into coherent units of content that belong together on a page, and each segment identified should be treated as a document unit. However, just like with most of the aforementioned problems, page segmentation, too, is still in its infancy.

For an optimization, the click behavior and dwell times on certain pages may be the best features to determine what parts should be indexed, whether pages should be merged into one, or one divided into many. Furthermore, such information on user behavior would be very useful for ranking results in the personal search. Currently, however, such behavioral data is probably not even available to commercial search engines.

5 RELATED WORK

WASP is directly related to prior work on desktop search, including the already mentioned Stuff I've Seen [6]. However, apart from not indexing all documents that may exist on a desktop, the intended usage differs slightly as well: WASP aims to track everything a user has seen, as they saw it, and in that sense provides some notion of versioning. While not yet implemented, a future version should explore the functionality once implemented in diff-IE, i.e., to rank pages that evolved differently from static ones, and this way provide immediate insight in changes of the web over time [13].

WASP is also related to search tools for web archives, such as ArchiveSpark [8]. However, due to handling a single user's view of the online world only, the system aspects to be addressed include less emphasis on scalability. Developments in the UI/UX of web archive search are, however, likely transferable, in both directions—as argued in Section 4.2, what we learn from observing interactions with personal web archives may very well carry over to the large web archives of interest to Digital Humanities researchers [3].

We find that a new blend of techniques that have been proposed previously will be necessary to design the right user experience, and we realize that we have only scratched the surface so far. For example, searching the social web is different from searching the web, as shown convincingly in [2]. We also highlight the immediate relevance of research into focused retrieval carried out in context of INEX. The question of how to determine a retrieval unit has clearly not been solved, yet, and the usage scenario of personalized web archive search that we envision has increased the urgency to revisit that line of research.

6 SUMMARY

This paper introduces WASP, a prototypical implementation of a personal web archive and search system, it provides a first qualitative evaluation of such a system, and outlines future steps in this regard, as well as discusses the challenges that such systems face. WASP combines state-of-the-art archiving and retrieval technology to which it adds an intuitive and tailored search interface. Generally, the use case for personal web archive search is more the one of a re-finding engine. We identify current limitations in archiving technology for this use case and discuss how the evaluation of a search engine has to be adapted for search in personal web archives (e.g., to several versions of a single web page when it is revisited). In the same context, we discuss what content should be archived and what content should be indexed, highlighting privacy issues (e.g., archiving in incognito mode) and advantages (re-finding information using only local data).

REFERENCES

[1] E. Adar, J. Teevan, and S.T. Dumais. 2008. Large scale analysis of web revisitation patterns. In *CHI '08*. 1197–1206.

[2] O. Alonso, V. Kandylas, S.-E. Tremblay, J.M. Hofman, and S. Sen. 2017. What's Happening and What Happened: Searching the Social Web. In *WebSci '17*. 191–200.

[3] Anat Ben-David and Hugo Huurdeman. 2014. Web Archive Search as Research: Methodological and Theoretical Implications. *Alexandria* 25, 1-2 (2014), 93–111.

[4] J. Shane Culpepper, Fernando Diaz, and Mark D. Smucker. 2018. *Report from the Third Strategic Workshop on Information Retrieval in Lorne (SWIRL 2018)*. Technical Report.

[5] E. Cutrell, D. Robbins, S. Dumais, and R. Sarin. 2006. Fast, Flexible Filtering with Phlat. In *CHI '06*. 261–270.

[6] S. Dumais, E. Cutrell, J.J. Cadiz, G. Jancke, R. Sarin, and D.C. Robbins. 2003. Stuff I've Seen: A System for Personal Information Retrieval and Re-use. In *SIGIR '03*. 72–79.

[7] C. Eickhoff, K. Collins-Thompson, P.N. Bennett, and S.T. Dumais. 2013. Designing Human-Readable User Profiles for Search Evaluation. In *ECIR 2013*. 701–705.

[8] H. Holzmann, V. Goel, and A. Anand. 2016. ArchiveSpark: Efficient Web Archive Access, Extraction and Derivation. In *JCDL '16*. 83–92.

[9] Milad Alshomary Benno Stein Matthias Hagen Martin Potthast Johannes Kiesel, Florian Kneist. 2018. Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment. *Journal of Data and Information Quality* (2018).

[10] W. Jones. 2010. *Keeping found things found: The study and practice of personal information management*. Morgan Kaufmann.

[11] H. Ma, R. Chandrasekar, C. Quirk, and A. Gupta. 2009. Improving search engines using human computation games. In *CIKM '09*. 275–284.

[12] Th. Samar, M.C. Traub, J. van Ossenbruggen, L. Hardman, and A.P. de Vries. 2018. Quantifying retrieval bias in Web archive search. *International Journal on Digital Libraries* 19, 1 (01 Mar 2018), 57–75.

[13] J. Teevan, S. Dumais, and D. Liebling. 2010. A Longitudinal Study of How Highlighting Web Content Change Affects People's Web Interactions. In *CHI '10*.