On the Development of a Collaborative Search System

Sindunuraga Rikarno Putra, Kilian Grashoff
Delft University of Technology
Delft, The Netherlands
{sindunuragarikarnoputra,k.c.grashoff}@student.tudelft.nl

Felipe Moraes, Claudia Hauff
Delft University of Technology
Delft, The Netherlands
{f.moraes,c.hauff}@tudelft.nl

ABSTRACT

Collaborative search is an active area of research in the IR community (and has been for many years)—despite this, there is a lack of open-source tools available to jump-start research in collaborative search. It is common for collaborative search researchers to implement their own tooling, leading to unnecessary duplicate engineering efforts. In this work, we describe the design process and challenges in implementing `SearchX`, an open-source collaborative search system, built using modern Web standards. `SearchX` implements essential features of collaborative search as found in the literature. In the design process, we focused on providing support for modern research needs (such as running crowdsourcing experiments and fast prototyping). We open-sourced `SearchX` https://github.com/felipemoraes/searchx-frontend (front-end) and https://github.com/felipemoraes/searchx-backend (back-end).

1 INTRODUCTION

Web search is generally seen as a solitary activity, as most mainstream technologies are designed for single-user search sessions. However, for a sufficiently complex task, collaboration during the information seeking process is beneficial [7]. A survey by Morris [14] has shown that collaborating during search is a common activity, albeit using ad hoc solutions such as email and instant messaging. Morris also found a significant increase in the number of people who collaborate during search at a regular basis, from 0.9% in 2006 to 11% in 2012. This increasing use of collaborative search (CSE) has also been reflected in the research community, where CSE has been an active area of research for many years. Workshops that explicitly focus on collaborative search—and more generally information seeking—have started to appear in 2008 [18] and continue to do so to this day, e.g. [2].

In contrast to single-user search where a number of up-to-date and open-source tools are readily available (e.g. Terrier[1] and Elasticsearch[2]), the CSE research community has currently just one maintained open-source option (`Coagmento`, cf. Table 1) despite the fact that researchers have designed and implemented a number of systems in the past ten years [1, 4, 8, 9, 15, 16, 21, 24]. While `Coagmento` provides an extensive collaboration feature set, it requires users to either install a browser plugin or an Android/iOS app, making it less viable for large-scale CSE experiments which are often conducted with crowd workers. Furthermore, we believe as researchers we should have a choice of tooling, instead of relying on a single one.

[1] http://terrier.org/
[2] https://www.elastic.co/

DESIRES 2018, August 2018, Bertinoro, Italy

For these reasons, we have designed and implemented `SearchX`, a CSE system built on modern Web standards, allowing it to be accessed from multiple platforms without the need for user-side installations. We designed `SearchX` specifically for CSE research and provide a comprehensive documentation to enable others to implement and run their own CSE experiments.

2 BACKGROUND

Collaborative search is a subset of the more generic field of collaborative information seeking (CIS). Golovchinsky et al. [10] have characterised the collaboration aspect of online CIS along four dimensions: *intent* (explicit or implicit), *mediation* (user interface or algorithm), *concurrency* (synchronous or asynchronous), and *location* (remote or co-located). Morris [14] suggested two additional dimensions: *role* (symmetric or asymmetric) and *medium* (Desktop or emerging devices). In our interpretation, collaborative search is scoped around collaboration of explicit intent, with the mediation and role dimension explored in designing the system, and the other three dimensions (concurrency, location, medium) describing the potential application scenarios of the system.

Systems for Collaborative Search. Our analysis of previously proposed systems in Table 1 is limited to those similar to `SearchX`—systems that support at least *synchronous* and *remote* collaborations. Additionally, we limit the scope to text retrieval systems, since it is the most common use case in Web search. One of the first attempts at such system was the design of `SearchTogether` [15], which focused on supporting awareness, division of labour, and persistence. Paul and Morris [16] built `CoSense`, an extension for `SearchTogether` to improve sense-making by providing additional views. Shah et al. [21] built upon the weaknesses of `SearchTogether` and created `Coagmento`, which has been analyzed for its experimental suitability in [11].

More recent systems were created to explore specific aspects of online collaborations. Golovchinsky et al. [9] designed `Querium` to better support the collaboration in an exploratory search process, specifically through implementing a shared document history that ranks documents based on relevance feedback. Capra et al. [4] designed `ResultsSpace` to study mostly asynchronous collaborations (though synchronous collaborations are possible too), therefore features for direct communication such as a chat were not added. Yue et al. [24] investigated the search behaviour of users and designed `CollabSearch` with basic collaborative features for their purpose. Leelanupab et al. [12] explored the effectiveness of visual snippets for sense-making by introducing the `SnapBoard` feature into the `CoZpace` system.

As stated before, most of the listed systems are only described in publications, and not open-sourced (or even available as binaries). As mediation is a vital factor for CSE, we first analysed how

Table 1: Feature comparison of existing remote collaborative search systems and SearchX (ordered by publication year of the first paper describing the system). A dash – indicates that this information is not available. Language and platform abbreviations: JS=JavaScript, BP=Browser Plugin, IE=Internet Explorer, FF=Firefox, GC=Google Chrome. Note that we only list programming languages listed in the respective papers (if no open-source code is available). †The Coagmento iOS app is only available in Apple's US app store.

	Search Together [15]	CoSense [16]	Coagmento [21]	Querium [9]	ResultsSpace [4]	CollabSearch [24]	CoZpace [12]	SearchX
Division of Labour								
Group Chat	✓	✓	✓	✓		✓	✓	✓
Document Sharing	✓		✓	✓				
Sharing of Knowledge								
Bookmarking / Document Saving			✓			✓		✓
Document Rating	✓	✓		✓	✓		✓	✓
Document Annotation	✓	✓	✓				✓	✓
Awareness								
Query History	✓	✓	✓	✓	✓	✓	✓	✓
Document History		✓	✓	✓				
Document Metadata	✓	✓	✓	✓	✓		✓	✓
Group Summary		✓		✓		✓	✓	
Colour Coding		✓				✓		✓
System Mediation	Split Search	–	–	Ranked Doc. History	Re-ranked Search Results	–	–	–
Tool Availability								
Functioning	✗	–	✓	–	–	–	–	✓
Open Source	–	–	✓	–	–	–	–	✓
Last Update	2009	–	2018	–	–	–	–	2018
Language	–	–	PHP & JS	JS	PHP	–	JS	JS
Platform	BP (IE)	–	iOS†, Android BP (FF, GC)	Web	Web	Web	Web	Web

mediation was designed in prior works, and used that as a starting point in implementing SearchX. As CSE solutions should require low additional effort compared to ad hoc solutions [9, 11, 14], we strove to implement features that look familiar to users (who all use Web search engines) today.

Designing Mediation. There are two main directions in developing mediation for CSE: *interface mediation* adapts the search interface towards a multi-user context, usually in the form of a shared workspace; *system mediation* directly mediates the collaboration process, mostly through re-ranking of documents [4, 9] or modifying the distribution of documents [15]. Both types of mediation are complementary to each other. The support for collaboration features can be categorized along three lines [6, 20]: *division of labour*, *sharing of knowledge*, and *awareness*. Table 1 provides a feature comparison of prior works [4, 9, 15, 16, 21, 24] in relation to these concepts. We now elaborate on each one and outline what is implemented in SearchX.

Division of Labour refers to the distribution of work load across collaborators. This division can be left to the user (user-driven) or mediated by the system. The latter can be implemented at the user level through assignment of roles, or at the document level through assigning different document subsets [22]. Prior systems mostly support user-driven division of labour through the provision of communication features. Group chat and document sharing (the explicit recommendation of a document to a collaborator) are two features which have been shown to be favoured by users [15, 21]. Following this, SearchX implements group chat; we argue that document sharing in the sense above can be achieved through the chat feature as well and thus does not warrant a separate UI element.

Sharing of Knowledge refers to the ability to share ideas and information effectively between collaborators [23]. This can be facilitated either through shared workspaces [19], or through the re-ranking of search results based on relevance feedback [5]. Prior systems support sharing of knowledge primarily through providing a shared workspace with features for collectively capturing information. Bookmarking documents (i.e. document saving) and document rating are both relevance feedback mechanisms, with prior systems either implementing one or the other. Document bookmarking promotes shortlisting, which involves forming and refining a shared list of potential resources [11]; document rating provides a finer granularity of feedback, which is needed for algorithmic mediation [4, 9]. Document annotation supports the previous two features by communicating the rationale behind an action [11, 15]. The choice of features largely depends on the experimental setup, therefore SearchX implements all three in a way that toggling individual features is easy. In SearchX document saving is instantiated as *bookmarking* as users are familiar with this concept.

Awareness is defined as *"the ability to maintain some knowledge about the situation and activities of others"* [13], encompassing knowledge of the workspace and collaborators' actions, as well as the ability to instantaneously notice changes on the work conducted. Prior systems focus on providing lightweight information regarding collaborators' search activities (query history, document history, and colour coding) and the overall sense-making process (document metadata and group summary). All features were found to be useful in past experiments, except for the document history which Kelly et al. [11] reported to provide too much information. As group summaries are mostly beneficial for asynchronous sessions [20], SearchX implements query history, document metadata, and colour coding as awareness features.

3 SYSTEM DESIGN

SearchX is designed for the following experimental workflow: a researcher first implements an experimental setup of their user study using SearchX (either relying on existing features, or adding their own). Each study participant accesses the SearchX instance through a designated URL; all major browsers are supported. The system then allocates a collaboration group to each set of $m \geq 2$ participants (m is a configuration parameter). Throughout a group's search session (which may include pre/post questionnaires), SearchX continuously captures fine-grained user activity logs.

We now discuss the architecture of SearchX and then elaborate on the two main design directions: supporting collaboration, and empowering research.

3.1 Architecture

When implementing SearchX, we chose to start from an existing system/interface to save development time. The options were limited as search engine interfaces are generally not open-sourced. We decided to use the single-user Pienapple search system [3][3] as a starting point, as it provides a generic Web search interface built with modern Web technologies (node.js[4], React[5]) which have active developer communities and are supported by large companies, ensuring that the system will be relevant technology-wise for the upcoming years. Given this base system, we vastly expanded its functionalities for collaboration and experimentation, and then refactored the code base to be modular and reusable.

SearchX's client-server architecture is shown in Figure 1. The front-end is responsible for presenting the interface, managing task sessions, and logging user activities; the back-end is responsible for communicating with the retrieval engine, and managing group creation and synchronisation.

Front-end. The front-end (shown in Figure 2) is developed using React (a JavaScript library). React manages its own data model, minimising communication with the back-end; it enforces the creation of standalone view components, resulting in an interface simple to modify and extend. As the front-end is a Web application, any user with a modern browser can access it without requiring additional installation.

The front-end consists of three logical abstractions. The *search interface* is composed of features related to searching and collaboration, and is presented to the user during the search session. Each feature is implemented as a standalone component which makes changing the layout or design of the interface efficient. Additionally, we separate the rendering component from the data management to make adjustments to the interface more efficient (e.g. adapting the interface for mobile or emerging devices). The *task session* implements the desired experimental setup, and controls the search task, group creation, and the experimental procedure (e.g. a pre-test, the search session, and then a post-test). An experimental procedure consists of a sequence of pages, which we bootstrapped by implementing template components for the search session, and questionnaires, which we found to be the most commonly required templates in our experiments. The logger accepts activity data from each component, and regularly sends the logs to the back-end through an HTTP request for storage. This abstraction provides a clear separation of concern between interface features, experimental setup, and data collection, making it clear which part of the system needs to be changed for a particular experimental need.

Back-end. The back-end is developed with the node.js server environment, which directly supports asynchronous I/O operations, making it suitable for applications requiring real-time updates. An added benefit of node.js is its language (JavaScript)—developing both the front-end and back-end in the same language made the development more manageable for us. The back-end provides the application data services which are made available to the front-end through APIs—implemented using the express[6] framework for HTTP and the socket.io[7] library for Web sockets. We chose these two libraries as they are currently the most common libraries for their respective role. We chose MongoDB[8] for the data storage as it uses a dynamic data schema, providing added flexibility during the development and modification of features.

The data services are categorised into four types. *Retrieval services* includes communication with the retrieval system through the provider, and further processing of the retrieval results through the regulator. Currently, we provide support for the Bing Search API for searching the Web, and support for Elasticsearch and Indri[9] servers for custom collections. *Session services* handles group communication and assigning search tasks to users. *Collaboration services* includes the back-end logic of collaborative features in the front-end. *Utility services* includes data collection tools such as the log collector which stores user logs received from the front-end, and the URL scrapper which scrapes all documents returned to the user. Additionally, we also have a URL renderer which makes it possible to load external Web pages inside our Web based system (think of a browser inside a browser), allowing us to implement the front-end document viewer. This offers the possibility to keep users inside the system at all times, allowing the system to log user interactions within the documents as well. Both the URL scraper and URL renderer utilise a headless browser via Puppeteer[10].

3.2 Supporting Collaboration

As can be seen in Table 1, only three prior systems implement system mediation, with each of them implementing a different type. In contrast, a number of interface mediation features (i.e. all features apart from system mediation) are popular across systems. SearchX also primarily supports collaboration through interface mediation, while facilitating custom implementation of system mediation through the addition of a regulator layer in the back-end. We now discuss each implemented feature. Figure 2 shows the UI of our interface with all features enabled.

Group Chat. Even though knowledge sharing is already facilitated through more specialised mediation features, direct communication is necessary for coordination and discussions. We opted for a familiar pop-up design where the chat window is always visible in the

[3] The authors kindly provided us with their source code.
[4] https://nodejs.org/
[5] https://reactjs.org/
[6] https://expressjs.com/
[7] https://socket.io/
[8] https://www.mongodb.com/
[9] http://www.lemurproject.org/lemur/
[10] https://github.com/GoogleChrome/puppeteer

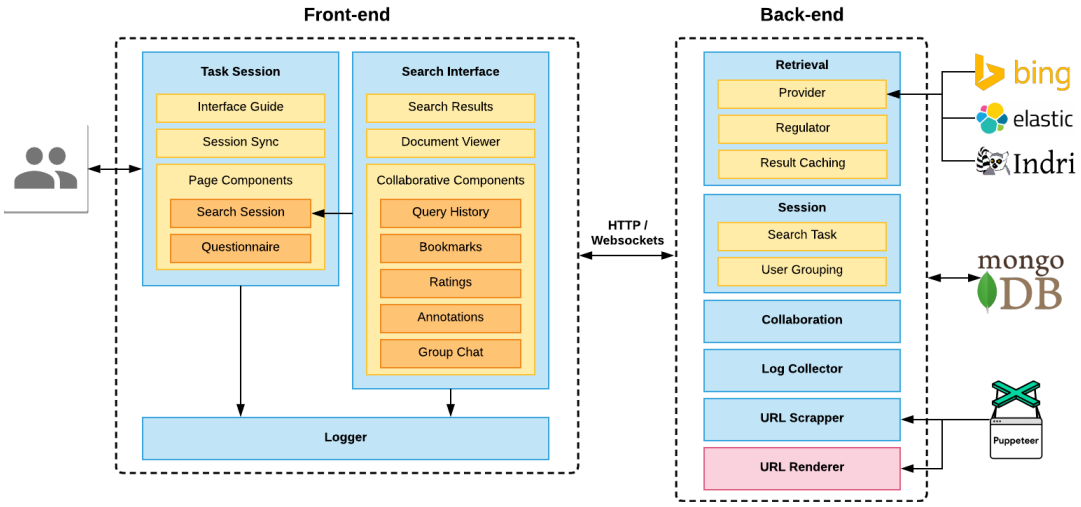


Figure 1: SearchX architecture overview.

interface but can be minimised when not in use (to avoid cluttering the interface). It is implemented using `converse.js`[11] which provides a robust chat window out of the box. A downside though is its black-box nature, which prevents direct access to the internals, making it difficult to extend (e.g. we are not able to automatically assign usernames). Currently the benefits still outweigh this limitation, but we will consider creating our own implementation in the future if needed.

Bookmarking. Apart from functioning as a means to bookmark documents for later revisits, document bookmarking also promotes the shortlisting strategy which involves curating a shared list of potential documents [11]. Given the central role of bookmarking in such collaborative efforts, we wanted to make it more accessible, therefore we implement the bookmark button directly next to each search result. The list of bookmarked documents is always visible in the sidebar to promote awareness of collaborators' actions; it is sorted by time, the most recent bookmarks appear at the top. In addition, users benefit in the sense-making process when given the option to manage and rearrange their bookmarks [11], therefore SearchX also implements pinned/starred bookmarks.

Document Rating. Document rating is mainly considered as fine-grained source of information for relevance feedback. To avoid cluttering of the search engine result page (SERP), we present the rating buttons not on the SERP, but inside the document viewer; the added benefit is that users can only rate once they have seen the document. Document rating is implemented as a like/dislike button to leverage users' familiarity with this type of interaction.

Document Annotation. Unlike existing systems, we implemented annotations as a message thread similar to chat interfaces. This setup highlights the bidirectional nature of the annotation process, promoting sense-making through the exchange of opinions. The annotation interface is presented inside the document viewer, directly next to the document to make adding new annotations a quick process. A feature planned for future development is the ability to highlight specific parts of the document along with an annotation.

Query History. This feature has been implemented in all prior systems; it provides awareness of collaborators' search activities, allowing users to avoid duplication of effort and be inspired by their collaborators' choice of keywords [15]. Its implementation is similar across prior systems: as a list of queries that can be clicked on to immediately open results for that query. SearchX provides a scrollable list of recent queries in the sidebar.

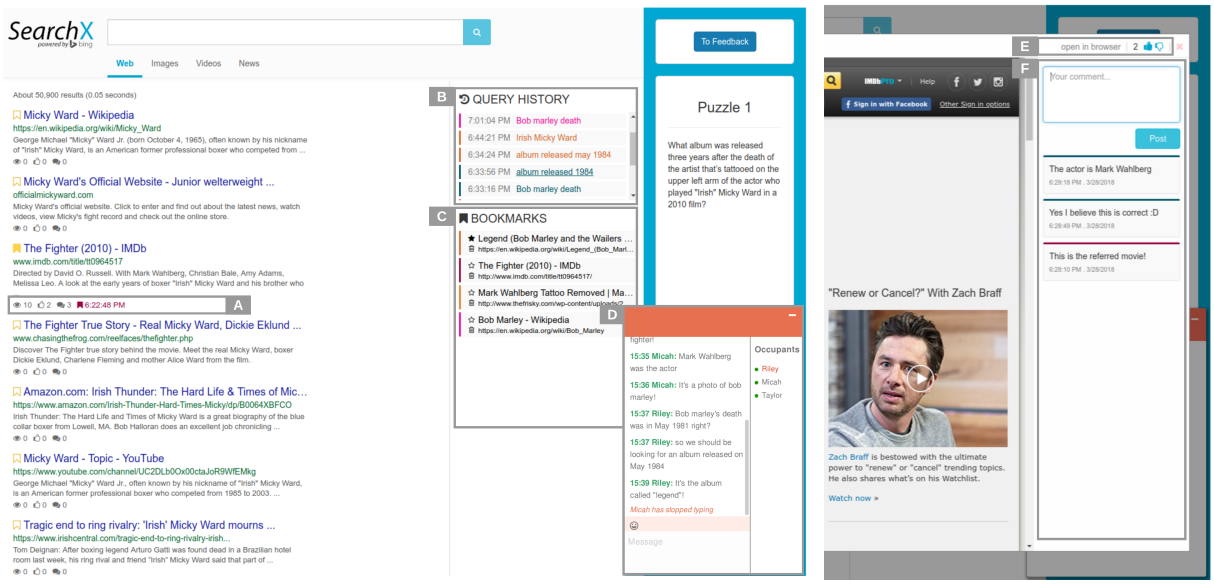
Document Metadata. This feature is presented below each SERP entry to provide information about collaborators' activities on the document. This allows users to quickly identify documents that are considered relevant by their group. The information is presented using a number of simple icons.

Colour Coding. We colour code elements of the interface that are associated with a particular collaborator's actions (such as querying and bookmarking). This allows users within the group to differentiate between the activities and contributions of each individual collaborator. The colours are generated randomly.

System Mediation. As stated before, SearchX does not implement a specific form of system mediation, but facilitates such an implementation if needed. In Section 2 we outlined that system mediation is usually performed in the form of modifications to the retrieved list of results (re-ranking). We have designed the retrieval service in the back-end to also contain a *regulator layer* that enables us to adjust the SERP sent to each collaborator based on the actions of the group's members.

The regulator layer collects the necessary input data for system mediation by fetching and aggregating it from the database. One example of such data is the current collection of bookmarked and up-voted results for the entire group, which can be used as input for relevance feedback. The input data can be sent to the search provider in order to incorporate the data in the retrieval algorithm. This is useful to incorporate features from the search provider into

[11] https://conversejs.org/



(a) Search result page

(b) Document viewer

Figure 2: SearchX collaborative search interface. [A] Document metadata, [B] shared query history, [C] shared bookmarks, [D] chat, [E] document rating, [F] document comments.

system mediation. For example, Indri supports relevance feedback by sending it a set of results. The input data can also be used directly in the regulator to re-rank or filter the list of results. This option can be used for e.g. distribution of labour by filtering the documents that are assigned to each user according to a distribution criterion.

3.3 Empowering Research

We now elaborate on how SearchX was designed as a tool for research.

Availability and Accessibility. SearchX is open-sourced for use and development by other researchers. We iterated on the installation process a number of times to make it simple and effective. We provide three example implementations of different experimental setups (synchronous collaborative search, asynchronous collaborative search, single-user search) to be expanded upon. We also put significant effort into extensively documenting how researchers can modify the system, e.g. by adding new UI features, or by changing the retrieval system in the back-end.

Study Creation. Currently, modifying the system requires programming knowledge as we do not provide a graphical interface to create user studies yet. However, we have created reusable implementations of common components in the experimental setup: questionnaires and the search session. The questionnaires are implemented using SurveyJS[12], which allows defining questionnaires directly using JSON. We have created a React component that abstracts over SurveyJS, adding logging features and flow control. We also did the same for the search session, which abstracts over the search interface, adding session-related logs, flow control, and a task bar to describing the search task. We found this to simplify the creation of new user studies, since it takes away much of the boilerplate code needed in configuring the experimental procedure.

Data Collection. A requirement for a CSE user study is the collection of user activity logs. In SearchX, we have added logging to all interactive components of the system so that it records when a user hovers over or directly interacts with a component (e.g. clicking, querying, opening a document). We also log session related data (e.g. starting/finishing the search session, submitting a questionnaire) and interactions with the browser (e.g. changing tabs), which helps understanding all actions executed by a user. All logs are captured directly by the interface without the need for third party plugins installed by the user. All logs are defined and implemented in the front-end, while the back-end only handles storage of logs, making it easy to modify the logs or create additional logs.

Interface Guide. Prior works report that some features of their system were not explored much by users because they do not know or understand it [9, 15]. We solve this issue by adding a guided interface walk-through of the interface (built using Intro.js[13]) which explains step-by-step what each feature is meant to do. This interface guide is launched when a user first starts the search session, ensuring that they are aware of the features we want them to use, before moving on to the search task.

4 CHALLENGES

We now discuss issues that are usually hidden from view—things that did not go as intended or slowed down the process. While the design decisions of SearchX were taken by the last two authors

[12] https://surveyjs.io/

[13] https://introjs.com/

of this paper, the engineering effort of making the design a reality was made by the first three authors (two MSc computer science students working on their thesis and a PhD student).

Iterating on the Experimental Setup. We initially implemented the basic version of SearchX with a paper deadline in mind. This led to a working but not very modular version of SearchX, which we realized when attempting to implement a number of CSE experiments—for each experiment, multiple files in both the front-end and back-end required changes. Since we wanted the system to be reusable for different experiments, we invested effort onto refactoring the code for a more intuitive experimental setup. We started fixing this in the front-end by separating out all code related to the experimental setup from the search interface and encapsulating them into reusable React components. While this simplified the experimental setup, the communication with the back-end remained a complex issue. We now limit the responsibility of the back-end to only group management and synchronisation, allowing us to directly implement the limited range of functionality inside the task components. If we would have spent more time on the initial design, we would have saved substantial development time, iterating a number of times on the architecture and the interactions among the components.

Deploying a Crowdsourced Study. As an effort to support online studies, we adapted SearchX for crowd-sourced studies. During a first CSE pilot on Crowdflower, we found crowd workers to not be overly motivated to properly execute our assigned collaborative search tasks (many tasks on Crowdflower tend to be short and do not require elaborate instructions such as image labeling). We found two ways around this issue: (i) a new platform and (ii) actively encouraging complying behaviour. We switched to the research-focused Prolific[14] platform which was shown to provide higher quality data [17]—something we found to be true as well in our work. We also spent significant development time on monitoring workers' attentiveness and actively keeping them on track. We logged browser interactions (change tabs, context menu) and notified workers about their tab changes in real-time (after n tab changes a worker is no longer paid). We also added quality control questions and disabled copy and paste operations in the questionnaires. All these steps improved the quality of data we collected, but were slow to be implemented as we discovered them as solutions to worker compliance issues one by one after running another (and another) pilot study.

Synchronising Group Sessions. Running a synchronous search session through a crowdsourcing platform is tricky, since workers are not available right away, therefore a type of "waiting room" is needed for the grouping so that workers assigned to a single group start their search session at the same time. This problem becomes particularly intense as the group size increases—an experiment with 20 workers requires 20 workers to accept the task at roughly the same time. Another issue we encountered was that workers were disconnected from the grouping process when the page was refreshed/closed during the waiting period, resulting in the worker not being able to continue the study. We currently just warn workers that attempt to refresh/leave the Web page running SearchX but a better way to handle (and entertain?) workers in the "waiting room" is needed to enable CSE experiments with large group sizes.

Implementing a Document Viewer. Ensuring that crowdworkers remain within SearchX (and otherwise rescind the payment) is a good way of ensuring compliance, but of course this idea breaks down when we want the workers to interact with the SERP (and click on links and view documents in another browser tab). We thus needed to implement a document viewer (again requiring valuable development time) that allows users to view the document within SearchX. This is rather straightforward for static resources such as text or images, however it is not possible to render another Web page directly inside SearchX because of CORS (cross origin resource sharing) restrictions. We thus had to render the URL in the back-end and pass the rendered HTML to an `iframe` in the front-end. This is an imperfect solution though since the resulting page is static with most interactive elements disabled, and at times the rendering is not perfect. We are still improving this aspect of SearchX. Currently, to alleviate the issue of imperfect renders, we add a button to open the web page in another tab.

Dynamic Result List and System Mediation. If the regulator layer of SearchX is used, the list of results is no longer a function of only the user's query, because it can change based on other input data as described in Section 3.2. In our experience this can lead to several challenges.

Data analyses that use the state of the SERP that a user observes is complicated in use of system mediation features. This is because the SERP that a user observed cannot be replayed based on their query, since it also depends on the other input data that the regulator layer made use. A possible solution for this problem is to store the search results that were displayed. We implemented a logger for each search result that is visible on the user's screen to facilitate our data analyses.

Another aspect that needs to be considered when implementing system mediation is the interaction of the mediation features with the search interface. A result list that is modified can lead to a jarring user experience, especially if the list updates in real time. We consider that it is better to apply changes to the SERP after a user initiates an action (e.g. a new query or page change); by combining the update due to system mediation with a user-initiated update of the SERP the user is not confused that the page changes. Another approach to prevent confusion is to indicate to users when results have been omitted or re-ranked and to give them the option to enable and disable mediation features. In this manner, users are given the autonomy to decide when system mediation features are useful.

5 CONCLUSIONS

We have presented SearchX, a collaborative search system whose design and implementation is an ongoing process, born out of the unmet need for an open-source CSE tool that can be deployed online without the need for additional installations (one of the main reasons for ruling out Coagmento for our purposes). The version we have just described provides a good starting point for online collaborative search research. Having a well-functioning and modular system is the basic requirement for the research we actually want to conduct with SearchX: large-scale (think tens or hundreds

[14] https://prolific.ac/

of users) collaborative search. We will continue to develop SearchX and by open-sourcing the system we hope that other researchers can benefit from it too.

ACKNOWLEDGEMENTS

This research has been supported by NWO projects LACrOSSE (612.001.605) and SearchX (639.022.722).

REFERENCES

[1] Saleema Amershi and Meredith Ringel Morris. 2008. CoSearch: A System for Co-located Collaborative Web Search. In *CHI '08*. 1647–1656.
[2] Leif Azzopardi, Jeremy Pickens, Chirag Shah, Laure Soulier, and Lynda Tamine. 2017. Second International Workshop On the Evaluation of Collaborative Information Seeking and Retrieval (Ecol'17). In *CHIIR '17*. 429–431.
[3] Martynas Buivys and Leif Azzopardi. 2016. Pienapple Search: An Integrated Search Interface to Support Finding, Refinding and Sharing. In *ASIST '16*. 122:1–122:5.
[4] Robert Capra, Annie T. Chen, Katie Hawthorne, Jaime Arguello, Lee Shaw, and Gary Marchionini. 2012. Design and evaluation of a system to support collaborative search. *ASIST* 49, 1 (2012), 1–10.
[5] Colum Foley and Alan F. Smeaton. 2009. Synchronous Collaborative Information Retrieval: Techniques and Evaluation. In *ECIR '09*. 42–53.
[6] Colum Foley and Alan F. Smeaton. 2010. Division of Labour and Sharing of Knowledge for Synchronous Collaborative Information Retrieval. *IPM* 46, 6 (2010), 762–772.
[7] Jonathan Foster. 2006. Collaborative Information Seeking and Retrieval. *Annual Review of Information Science and Technology* 40, 1 (Dec. 2006), 329–356.
[8] Gene Golovchinsky, John Adcock, Jeremy Pickens, Pernilla Qvarfordt, and Maribeth Back. 2008. Cerchiamo: a collaborative exploratory search tool. *Computer Supported Cooperative Work* (2008), 8–12.
[9] Gene Golovchinsky, Abdigani Diriye, and Tony Dunnigan. 2012. The Future is in the Past: Designing for Exploratory Search. In *IIIX '12*. 52–61.
[10] Gene Golovchinsky, Jeremy Pickens, and Maribeth Back. 2008. A taxonomy of collaboration in online information seeking. *JCDL Workshop on Collaborative Information Retrieval* (2008).
[11] Ryan Kelly and Stephen J. Payne. 2014. Collaborative Web Search in Context: A Study of Tool Use in Everyday Tasks. In *CSCW '14*. 807–819.
[12] Teerapong Leelanupab, Hannarin Kruajirayu, and Nont Kanungsukkasem. 2015. Snapboard: A Shared Space of Visual Snippets - A Study in Individual and Asynchronous Collaborative Web Search. In *Information Retrieval Technology*. Springer International Publishing, 161–173.
[13] Olivier Liechti. 2000. Awareness and the WWW: An Overview. *SIGGROUP Bulletin* 21, 3 (Dec. 2000), 3–12.
[14] Meredith Ringel Morris. 2013. Collaborative Search Revisited. In *CSCW '13*. 1181–1192.
[15] Meredith Ringel Morris and Eric Horvitz. 2007. SearchTogether: An Interface for Collaborative Web Search. In *UIST '07*. 3–12.
[16] Sharoda A. Paul and Meredith Ringel Morris. 2009. CoSense: Enhancing Sensemaking for Collaborative Web Search. In *CHI '09*. 1771–1780.
[17] Eyal Peer, Laura Brandimarte, Sonam Samat, and Alessandro Acquisti. 2017. Beyond the Turk: Alternative platforms for crowdsourcing behavioral research. *Journal of Experimental Social Psychology* 70 (2017), 153 – 163.
[18] Jeremy Pickens, Gene Golovchinsky, and Meredith Ringel Morris. 2009. Proceedings of 1st International Workshop on Collaborative Information Seeking. *CoRR* abs/0908.0583 (2009).
[19] Steven Poltrock, Jonathan Grudin, Susan Dumais, Raya Fidel, Harry Bruce, and Annelise Mark Pejtersen. 2003. Information Seeking and Sharing in Design Teams. In *2003 International ACM SIGGROUP Conference on Supporting Group Work (GROUP '03)*. ACM, 239–247.
[20] Chirag Shah and Gary Marchionini. 2010. Awareness in Collaborative Information Seeking. *ASIST* 61, 10 (2010), 1970–1986.
[21] Chirag Shah, Gary Marchionini, and Diane Kelly. 2009. Learning Design Principles for a Collaborative Information Seeking System. In *CHI EA '09*. 3419–3424.
[22] Laure Soulier and Lynda Tamine. 2017. On the Collaboration Support in Information Retrieval. *Comput. Surveys* 50, 4, Article 51 (Aug. 2017), 34 pages.
[23] Ke-Thia Yao, Robert Neches, In-Young Ko, Ragy Eleish, and Sameer Abhinkar. 1999. Synchronous and asynchronous collaborative information space analysis tools. In *1999 International Workshops on Parallel Processing*. IEEE, 74–79.
[24] Zhen Yue, Shuguang Han, and Daqing He. 2012. Search tactics in collaborative exploratory web search. In *HCIR 2012*.