Tool Support for Knowledge Foraging, Structuring, and Transfer During Online Sensemaking

Michael Xieyang Liu

CMU-HCII-23-105
August 2023

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Brad A. Myers, HCII, CMU, Co-chair
Aniket Kittur, HCII, CMU, Co-chair
Kenneth Holstein, HCII, CMU
Daniel M. Russell, Google, Inc.

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Sensemaking, Decision Making, Knowledge Reuse, Developer Support Tools, Intelligent User Interfaces, Human-Computer Interaction.

Abstract

While modern search engines are excellent resources for finding information on the web, in order to put together that information into a useful mental model for learning or making a decision – such as picking a new car or choosing a JavaScript library – people often need to collect information about the options available and the criteria on which to evaluate the options, synthesize such information from various sources into a meaningful structure, and share and justify the results with others. This sensemaking process, often highly iterative and cyclical, puts a significant cognitive burden on users, and often requires them to externalize their evolving mental models rather than keeping everything in their working memory. However, the tools that people use for externalization – such as browser tabs, documents, spreadsheets, or note-taking apps – poorly support the constant shifts between collecting, extracting, organizing, and reorganizing that are needed. Worse yet, even if people do put in the work to externalize and share a summary of their sensemaking outcome (such as creating a list of suitable cars or a table of front-end libraries), it can still be difficult for subsequent users to evaluate whether they can or should trust and reuse that work so that they don't have to start from scratch.

In this thesis, I aim to bridge the gap between the rapidly evolving mental models in peoples' heads and the externalization of those models. Specifically, I design and build interactive systems *to reduce the costs and increase the benefits of externalization*, thereby capturing more of the cognitive work that users engage in while making sense of information in order to help them as well as subsequent people who might benefit from their work.

To help the initial users collect and structure information, this thesis first describes **Unakite**, a browser extension that enables people to easily collect and organize information into a comparison table in a sidebar as they are searching and browsing, which significantly lowered the friction of externalizing mental models compared to conventional approaches like taking notes and saving screenshots in a separate Google Doc. In addition, the knowledge captured in the comparison tables helped subsequent users better understand previous authors' sensemaking processes and rationale. Building on Unakite, we explored approaches to further reduce the cost of externalization and help people focus on their main activity of reading and making sense of web content, such as by intelligently keeping track of key information and evidence on behalf of a user (the **Crystalline** system) and leveraging novel lightweight interaction techniques (the **Wigglite** system). To help subsequent users explore and evaluate previous users' work, I developed both a framework and the **Strata** system that collects and visualizes key signals about the context, trustworthiness, and thoroughness of previous design decisions and rationale. Finally, after gaining a deeper understanding of people's information needs and processes when sensemaking, I circle back to the beginning to help people with reading and understanding information based on those needs. Through the **Selenite** system, I provide people with a top-down comprehensive overview of the information landscape, enabling users to jumpstart their sensemaking processes and receive guidance for future exploration.

The series of work introduced in this thesis points to the importance of having tool support that helps users efficiently organize and manage information as they find it in a way that could also be beneficial to others, and therefore bootstrapping the virtuous cycle of people being able to build on each other's sensemaking results, fostering efficient collaboration and knowledge reuse.

Acknowledgements

This thesis would not be possible without my advisors, Brad Myers and Niki Kittur, who have been tremendously supportive over the past 6 years. They have provided guidance, resources, and, most importantly, the freedom that helped me grow into a more mature researcher where I am able to define and execute new research agendas. Brad and Niki also served as role models to me for their passion and enjoyment of research and their investment and pride in their students' successes.

I would also like to thank my other committee members, Ken Holstein and Dan Russell, for their valuable feedback and continued support. I also thank those who have been mentors and champions for me at different stages of my academic career: Jia Deng, Walter Lasecki, Danai Koutra, Dustin Smith, Todd Kulesza, Sarah D'Angelo, Advait Sarkar, Carina Negreanu, Ben Zorn, Andrew Gordon, Jeff Nichols, Titus Barik, and many more.

This thesis has received generous financial support from corporate and governmental sponsors, including the National Science Foundation, Office of Naval Research, Google, and Bosch. I also want to thank the anonymous study participants and paper reviewers who contributed to this work.

I am also thankful for the collaboration and support from wonderful friends and colleagues: Amber Horvath, Andrew Kuznetsov, Angel (Alex) Cabrera, Anhong Guo, Cassie Cao, Cynthia Ouyang, Cori Faklaris, Daye Nam, Eric Yiyi Wang, Felicia Ng, Gierad Laput, Haitian Sun, Haojian Jin, Han Zhang, Helen Tsui, Hong Shen, Hyeonsu Kang, Jiachen Wang, Jiannan Li, Jianzhe Gu, Jinding Xing, Jinlei Chen, Julia Cambre, Julia Qian, Karan Ahuja, Kexin Bella Yang, Lea Albaugh, Lynn Kirabo, Liang He, Mary Beth Kery, Meng Xia, Nur Yildirim, Orson Xuhai Xu, Qian Yang, Queenie Kravitz, Ruotong Wang, Saiganesh Swaminathan, Samantha Reig, Sherry Tongshuang Wu, Shiyan Yan, Sitong Wang, Stephanie Valencia[2], Tianshi Li, Toby Jia-jun Li, Wesley Hanwen Deng, Wode "Nimo" Ni, Xiaoyi Zhang, Xinci Weng, Xu Wang, Yan Chen, Yang Zhang, Yi Zhou, Yijun Hou, Yulan Feng, Zheng Yao, Zhihang Dong, Ziyan Wang, and many more.

I would like to express special gratitude to Tianying Chen, who has been a dear friend and by my side through numerous ups and downs over the past 6 years; Franklin Mingzhe Li, who not only shares the same exact birthday with me[1] but also has been my biggest champion throughout this Ph.D. journey; Yongsung Kim, who has been nothing but a north star in guiding me through the latter half of my Ph.D. ordeal. I would also like to thank Chen Yifaer (and her kitty Xiaode) — her melodious and calming singing voice has consistently been a source of solace for me during numerous long nights of self-doubt.

Finally, I am deeply grateful to my family — my parents, Weigang Li (李爲崗) and Jun Liu (劉軍), and my grandparents, Jin Li (李進), Xuefen Rong (榮雪芬), Yongqing Liu (劉永慶), and Zhiying Su (蘇芝英), for their unwavering love, support, and encouragement for the past 28 years. Their sacrifices and constant belief in me have been the driving force that pushes me forward.

[1] June 12, 1995

To the selfless, tenacious, and wise-minded.

Contents

List of Figures

List of Tables

List of Acronyms

Following a tradition[2] started by one of my doctoral advisors Brad Myers, the names of most of the systems described in this dissertation are acronyms based on gemstones or rocks. Below is a list of these acronyms and what they stand for, as well as references to the corresponding chapters in this dissertation.

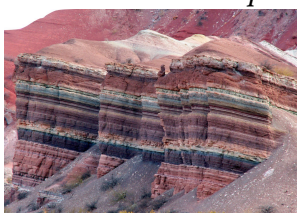
1. Unakite
   Users **N**eed **A**ccelerators for **K**nowledge for **I**mplementations in **T**echnology **E**nvironments
   A Chrome extension that enables people to collect, organize, and keep track of information about decision trade-offs and build a comparison table, which can be saved as design rationale for later use. (Chapter 3)

2. Crystalline
   Clipping **R**esulting in **Y**our **S**tructure as **T**ables **A**nd **L**ists **L**inked to **I**mplicit **N**otetaking **E**asily
   A system that automatically helps users collect and organize information by leveraging natural language processing of web content and passive behavioral signals that people naturally exhibit while searching and browsing. (Chapter 4)

[2] The full list of all systems and their acronyms from Brad and his students is available at `https://www.cs.cmu.edu/~bam/acronyms.html`.

3. Wigglite[3]
   Wiggling for **I**nformation **G**athering and **G**enerating **L**ightweight **I**mpressions for **T**riage and **E**ncoding
   A system that implements a novel interaction technique called "wiggling" for collecting information and encoding mental judgment of that information. Wiggling involves rapid back-and-forth movements of a pointer or up-and-down scrolling on a smartphone, which can indicate the information to be collected and its valence, using a single, lightweight gesture that does not interfere with other interactions that are already available. (Chapter 5)

4. Strata
   Sidebar **T**owards **R**euse and to **A**ssess **T**rustworthiness and **A**pplicability
   A framework and a system that collects and visualizes key signals about the context, trustworthiness, and thoroughness of previous design decisions and rationale to help subsequent people evaluate and reuse previous people's sensemaking results. (Chapter 6)



5. Selenite
   Smart **E**nvironment for **L**ogical **E**xtraction and **N**avigation of **I**nformation using **T**echnological **E**xpertise
   A novel system that provides people with a top-down comprehensive overview of the information landscape, enabling them to kickstart their sensemaking processes and receive guidance for future exploration. (Chapter 7)

[3]Note that Wigglite is NOT an acronym for a real gemstone. The image below was actually generated by DALL-E 2 using the prompt "a piece of rock that is called Wigglite, which shows lots of wiggles on the surface." I sincerely hope Brad is OK with this.

Chapter 1

Introduction

While modern search engines are excellent resources for finding information on the web, in order to put together that information into a useful mental model for learning or making a decision – such as picking a new digital camera, researching a medical diagnosis, or choosing a JavaScript library to build websites – people often need to collect information about trade-offs from multiple sources, extract and synthesize snippets of information into meaningful structures, or keep track of, share, and justify their decisions and rationale with others or to their future selves.

During this process of sensemaking, people's mental models constantly evolve as they gather more information about the decision space – the contexts relevant to their goals, the options available, and the criteria or constraints on which to evaluate the options. For example, a YouTuber seeking to upgrade her vlogging setup may learn about many different camera options from various websites. As she discovers these options, she prioritizes in her head which one she wants to investigate first, looking for video samples and reviews that speak either positively or negatively about those cameras in terms of various dimensions such as sensor resolution, zoom range, color accuracy, battery life, etc. Indeed, estimates suggest that up to 1/3 of the time people spent online [183, 245, 301], or around 24 billion hours per year (as of 2009) in the US alone [34], are spent performing such complex and cognitively-demanding sensemaking tasks.

This highly iterative process puts a significant cognitive burden on users, and often requires them to **externalize** their evolving mental models. However, this can be challenging for a number of reasons. First, people are generally sensitive to the cost structure of the tools they use for making sense of information, and any tool that adds a perceived burden or obstacle to their natural sensemaking process is not likely to be adopted. Second, people are often uncertain about which information will eventually turn out to be relevant, valuable, and worth capturing, especially at the early stages of their learning and exploration when they are overloaded with information.

Third, the need for externalization of their mental models is often not discovered until part way through an investigation process, such as realizing one is juggling many more options, criteria, and trade-offs than initially anticipated, or simply being required to document a decision and rationale for downstream auditing purposes, etc.

In addition, the tools that people have access to nowadays for externalization – such as browser tabs, documents, spreadsheets, or note-taking apps –- poorly support the constant shifts between information collecting, extracting, organizing, and reorganizing that are needed. Therefore, *the gap between the rapidly evolving mental models in peoples' heads and the lagging representations of those models in external documents* means that people often abandon their efforts to externalize halfway or avoid doing so in the first place [71, 75, 145, 232], and instead try to keep everything in their working memory, which, unfortunately, is not unlimited [159, 247, 337].

Furthermore, even if people do put in the work to organize and share an external representation of their learning outcome or decision (such as creating a list of reasonable vlogging cameras or a comparison table of front-end libraries), *it can be difficult for subsequent users to evaluate whether they can or should **reuse** that work.* Often, the perceived effort of deciding to reuse someone else's sensemaking knowledge might exceed that of redoing the sensemaking from scratch – it involves, for example, judging whether the initial knowledge creator's goals and context match with that of the subsequent user, and if the creator has the proper expertise and has performed a thorough job of surveying the information landscape. For example, a subsequent vlogger can theoretically take advantage of the camera options and reviews that the previous YouTuber has found, but they may have a vastly different budget to begin with, and may need to double-check if there are additional alternatives available on the market, etc., and may ultimately prefer to redo their own sensemaking despite of the rich knowledge that the first person had summarized.

1.1 Thesis Statement and Overview

To address the challenges mentioned previously, I designed, implemented, and studied a series of intelligent systems in this dissertation. The thesis for this dissertation is:

I design and build user interfaces, interactions, and computational scaffolds that enable initial users to more efficiently and effectively read, comprehend, collect, and organize information to make and justify decisions, while automatically capturing the sensemaking context to help subsequent people understand and evaluate those decisions.

The research efforts described in this thesis can be divided into four interrelated stages: *helping the initial user (1) find, (2) collect, (3) structure information, and helping subsequent users (4) evaluate and reuse the initial users' sensemaking results.* I[1] posit that *the same signals that could help computational tools understand initial people's context and intents can also help subsequent people evaluate and decide whether and how to reuse the artifacts scaffolded by those tools.* Therefore, an overarching goal for this thesis is to understand and explore synergies in making it worth people's effort to provide rich signals about their sensemaking context and mental models to external systems that can help them more effectively find, collect, organize, and refactor information and knowledge, and leveraging those signals to lower the cost for subsequent users to evaluate and reuse the knowledge externalized by previous users.

To ground the research, I primarily focus on two domains: programming and consumer decision-making, because both involve frequent learning and research on the web followed by potentially impactful decisions, and are specific enough to engage with deep contextual complexities, yet provide insights that generalize to other sensemaking activities. In the next section, I will provide a brief overview of the structure of this thesis.

1.2 Thesis Outline

In the rest of this dissertation, I first provide an overview of the background and related work (**Chapter 2**). Then I introduce five novel intelligent interactive systems that are designed to support the whole range of stages in sensemaking mentioned previously. These systems are:

Unakite: Scaffolding in-situ Information Collection and Organization (Chapter 3)

Knowledge workers spend a significant portion of their time searching for solutions to their problems online. While numerous tools have been developed to support this exploratory process, in many cases, the answers to their questions involve trade-offs among multiple valid options and not just a single solution. Through interviews, I discovered that people express a desire for help with decision-making and understanding trade-offs. Through an additional content analysis study, I observed that many answers describe such trade-offs. In addition, the trade-offs can often be captured in a tabular format, comparing different *options* (items of comparison) with respect

[1]Although this thesis is based on research projects that I have personally led, this document predominantly contains the pronoun "we" out of respect to all of my collaborators who have contributed to the research.

to different *criteria* (dimensions used by people to compare the options) using *evidence* (snippets of information) that people found online.

To lay the foundation for people to capture and organize information about these trade-offs, I designed the **Unakite** system, which enables people to collect, organize, and keep track of information about decision trade-offs and build a comparison table, which can be saved as design rationale for later use. In our evaluation, Unakite significantly lowered the friction of externalizing mental models compared to conventional approaches like taking notes and saving screenshots in a separate Google Doc; in the meantime, the knowledge captured in the comparison tables helped subsequent users better understand previous authors' sensemaking process and rationale.

Crystalline: Automating Information Collection and Organization (Chapter 4)

My previous system Unakite encouraged authors to document their decision-making processes and results using the tool's lightweight collecting and organizing features. However, it remains a laborious process for people to manually identify and clip content, maintaining its provenance and synthesizing it with other content. To address this potentially high interaction and interruption cost of manually collecting and organizing information, I designed and built the **Crystalline** system, which explored automatic approaches by leveraging natural language processing (NLP) and passive behavioral signals that people naturally exhibit while searching and browsing such as mouse movement and dwell time. Our lab study suggests that, using Crystalline, people are able to create Unakite-style comparison tables faster with a significantly less operational cost, without sacrificing the quality of the tables.

Wigglite: Lightweight Gestures for Collection and Triage (Chapter 5)

To incorporate additional user control back into the information collection and organization process, I explore interactions that can facilitate the seamless transfer of a user's internal mental judgments to an external system. While it is a challenge for the cost of this transfer to be zero, I aim to reduce the overhead significantly by exploring a new interaction technique called "wiggling," which can be used to fluidly collect, organize, and rate information during early sensemaking stages with a single gesture. Wiggling involves rapid back-and-forth movements of a pointer or up-and-down scrolling on a smartphone, which can indicate the information to be collected and its valence, using a single, lightweight gesture that does not interfere with other interactions that are already available. Through implementation and user evaluation, I found that wiggling

helped participants accurately collect information and encode their mental context with a 58% reduction in operational cost while being 24% faster compared to a common baseline.

Strata: Evaluating and Reusing Information and Knowledge (Chapter 6)

As the amount of information online continues to grow, a correspondingly important opportunity is for individuals to reuse knowledge that has been summarized by others rather than starting from scratch. However, appropriate reuse requires judging the relevance, trustworthiness, and thoroughness of others' knowledge in relation to an individual's goals and context. Through an analysis of prior research on sensemaking and trust, along with new user interviews, I synthesized a framework for reuse judgments (Table 6.1). From this framework, I developed a set of techniques for capturing the initial decision maker's behavior and visualizing signals calculated based on the behavior, to facilitate subsequent consumers' reuse decisions, instantiated in a prototype system called Strata. Results of a user study suggest that the system significantly improves the accuracy, depth, and speed of reusing decisions. These results have implications for systems involving user-generated content in which other users need to evaluate the relevance and trustworthiness of that content.

Selenite: Grounded Reading and Comprehension (Chapter 7)

Previous chapters in this thesis and past research have generally assumed that users have already discovered and comprehended the information they chose to integrate into the system. However, through formative interviews and inquiries with participants who recently explored unfamiliar topics and domains, we discovered that people frequently grapple with finding, reading, understanding, and navigating the information in the first place, largely due to a lack of comprehensive overview of the information space. This deficiency arguably hinders the subsequent development and refinement of their mental models.

Therefore, in this chapter, we introduce a novel system named Selenite that provides users with a comprehensive overview of the information space upfront to jumpstart as well as guide their subsequent reading and sensemaking processes. Through a performance evaluation of Selenite, we verified its ability to provide a sufficiently accurate and high-quality global overview to the users. Furthermore, a usability and case studies revealed that Selenite significantly accelerated and improved users' information reading and comprehension processes.

In conclusion, I reflect on the contributions made and valuable insights gained throughout this dissertation, and provide a glimpse into potential avenues for future research (**Chapter 8**).

Chapter 2

Background & Related Work

Sensemaking is widely considered to be the process of searching, collecting, and organizing information to iteratively develop a mental model of an information space in service of a user's goals [288, 307]. A number of models of sensemaking have been proposed, including Russell et al.'s cost structure view [307], Dervin's sensemaking methodology [98], Klein et al.'s data-frame model [195], organizational process views [91, 126], organizational adaptation views [91, 258], and the notional model by Pirolli and Card [287]. At a high level, these models agree that a sensemaking process involves alternating between two phases: **foraging**, which involves people searching for and extracting information, often from various data sources; and **structuring**, the process of integrating the amassed information to form a schema or representation to interpret the space [288]. More recent work has also explored the concept of distributed sensemaking [114,219], which involves **evaluating** whether to take advantage of and reuse an initial user's decision making results or sensemaking artifacts [114,136,318] and **adapting and reusing** them for their own purposes [137, 284]. Below, I briefly discuss the related work for each of these stages.

2.1 Foraging Information

Prior work has reported that the foraging phase, which involves collecting and extracting information, is where people tend to spend the majority of time during a sensemaking process [55,69,244,287]. Thus, there have been many research and commercial tools that try to help people better capture information during this phase. Some focused on keeping track of entire webpages or documents, such as SenseMaker [37], browser bookmarks, and reading lists; while others

enabled users to capture finer-grain units within a web document, such as Hunter Gather [315], Clipper [193], and Google Notebook [133].

However, foraging can be challenging both *physically*, because the complex structure of web pages can make it hard to select the desired content with exact boundaries and tiresome to repeat the selection frequently [72,78,303], and *cognitively*, because in early stages of sensemaking people often are not sure what will actually end up being relevant, useful, and worth collecting [41,114].

With respect to the physical demand for collecting and extracting information while searching and browsing, prior work has pointed out that users need quick and lightweight interaction techniques, because if the physical cost is too high (such as specifying the boundaries of some desired content, copying it, switching context to the target tab or window, and transferring the information into the application where it will be stored [113]), users tend not to capture information in the first place [159, 232, 247, 337]. Prior work has explored various ways to lessen the physical cost – on the one hand, multiple approaches have been proposed to make selecting desired content faster by offering pre-defined selection boundaries. For example, systems like Entity Quick Click and Citrine [47,174,330] employ techniques like named-entity recognition [243] to pre-process and highlight semantically meaningful entities in a document and allow users to collect and annotate relevant information with a single click. On the other hand, research and commercial products have explored collecting information on behalf of users as they search and browse the web. For example, works such as Thresher [160] and Dontcheva et al.'s web summarization tool [101] let users create and curate patterns and templates of information that they want to collect through examples, and then automatically collect that information from pages that users visit in the future.

With respect to the cognitive cost for collecting and extracting information in-situ, people often have to reason and make a decision about which and how much information to capture despite being uncertain about its future value [67, 72, 151, 193, 372]. In addition, such frequent mental context switches away from reading and making sense of the actual web content can be extra interruptive [159,193,312]. Before a user has built a good mental model of an information space, they have to manage the tension between extracting too much information that later turns out to be irrelevant, versus extracting too little information and later having to revisit webpages to collect additional information. Recent work by Chang et al. [72] proposed one potential way of easing the cognitive burden by allowing users to just create "fuzzy" selection of web content on the go and defer the precise specification of what to capture and persist till a second pass.

2.2 Structuring Information

After collecting and extracting useful information, a user needs to synthesize it into structures that are useful for interpreting the information space and achieving their goals of learning or decision making. The idea of building structured representations of information has a history dating back at least to the visions articulated by Vannevar Bush and others of associative memory, spatial hypertext, and other means of extending the human intellect (e.g., [64, 105, 227, 248, 268]). Since then, there have been many attempts at structuring web content, including lists, tables, graphs, trees, mind maps, argument maps, and panels [363], however, empirical research has found that using such tools can feel like "learning a new language" [191].

Prior work has explored various ways of incorporating lightweight information classification into the foraging phase to leave clues and hints that scaffold later organization. For example, Clipper [193] and Adamite [163] all prompt the user to optionally categorize an information clip after it has just been captured. Spar.tag.us [161] enables users to associate custom tags with individual paragraphs. ForSense [293] leverages natural language processing to automatically cluster information clips based on themes and topics.

There have also been a number of research tools developed to support in-depth organizing and structuring. These include the WebBook and WebForager by Card et al. [68], which use a book metaphor to find, collect, and manage web pages; Butterfly by Mackinlay et al. [240] aimed at accessing articles in citation networks; the Navigational View Builder [264] which combined structural and content analysis; Elastic Windows, which provided information overview and location context [180]; Webcutter, which collects and presents URL collections in tree, star, and fisheye views [239]; SenseMaker [37] for evolving collections of information; PadPrints [157], a zoomable history browsing interface; and Scatter/Gather [90], a text-clustering interface for iteratively navigating through document collections. Related tools that support aspects of structuring include extraction pattern approaches such as in [101], Stepping Stones and Pathways [92], Cat-a-Cone [153], Data Mountain [298], TopicShop [29, 339], Hunter Gatherer [315], Haystack [182], and Internet Scrapbook [331].

Despite these attempts, the dynamic and evolving nature of sensemaking – particularly in the early stages – means that users often would avoid the cost of structuring or even committing to a particular structure. Even if they do organize information, the structures that they created often become obsolete as their mental representations evolve over the course of their investigation (such as realizing a particular criterion should be prioritized, which prompts an entirely different investigation of several new options, etc.), with no single type of structure likely to remain the most appropriate throughout the whole sensemaking process [114,152,193]. Instead, people often

would try to keep everything in their working memory, which, unfortunately, is not unlimited [50, 244, 302].

2.3 Evaluating and Reusing Information

Information and knowledge reuse has become a highly consistent paradigm across a wide range of fields and disciplines to advance their respective frontiers, such as reusing previous engineering best practices on future generations of products [42, 43], taking advantage of schemas and results from previous sensemaking episodes to create new representations and understandings of the world [114, 192, 262, 283], and plugging in previously written and well-maintained design patterns and code snippets to build novel software features and functionalities [4, 18, 122, 123, 200]. Reusing proven information and knowledge promises the benefits of potentially reduced workload and development cycles [42, 200], improved quality and performance [123, 142, 355], and more time for creation and innovation [171, 242, 246, 355].

Despite these benefits, consuming someone else's sensemaking results can incur significant costs, including deciding whether the initial user's context is similar enough to their own for that work to be relevant, and evaluating whether the initial user's trustworthiness and thoroughness are sufficiently high to believe in their results. Prior work has reported various factors that influence the *evaluation* of others' work, including but not limited to: domain name and URL, presence of a timestamp showing that the information is current or sufficiently up-to-date, authors' identification and indication of their expertise on the topic, citations to references or scientific evidence, and user ratings or reviews [26, 53, 108, 116, 125, 219, 251, 253, 255, 323, 341, 353]. These factors are common across a wide range of reuse scenarios such as choosing software architectures, libraries, and APIs, purchasing consumer products, handing-off or taking over design and management projects, etc. [39, 73, 142, 232, 246, 259, 318, 320, 327].

However, in reality, it has been repeatedly shown that people are often under-prepared and have trouble determining how to evaluate others' work [32, 251, 254, 313], which is often deemed to be too much effort [251, 318], having a high possibility of missing important details [253, 255]. As a result, users may end up starting from scratch rather than engaging in the potentially costly consumption of someone else's work if they are uncertain about how relevant and useful that work will end up being [114, 228, 229, 253].

Over the years, many systems have been developed to support knowledge hand-off and reuse, during which the current sensemaker (subsequent user) needs to make sense of and evaluate the appropriateness of reusing the results generated by a previous sensemaker (initial user) [246, 318].

Various metadata and properties parallel to the main artifacts of sensemaking have been proposed that would help subsequent users with this process, such as the awareness of the previous sensemaking process [102,283] (e.g., search queries and visited web pages), the level of expertise of the initial user [246,320], the context of the original sensemaking problem [246], and the initial user's design rationale [207, 208, 325]. However, it is both time and effort intensive for a sensemaker to keep track of their rationale and processes with little immediate payoff, which is also often for the benefit of others rather than themselves [232]. Even in situations where authors have the explicit wish to help, they are often uncertain of what metadata and properties to provide and how those can be instantiated using concrete signals that would be valuable to the consumers in evaluating the reusability of their sensemaking results [318].

In conclusion, prior research and existing systems have undoubtedly made substantial strides in assisting individuals in making sense of online information. However, despite these advancements, there still exists a notable amount of friction and cost when it comes to foraging and structuring raw data into synthesized knowledge, along with evaluating the appropriateness of reusing such knowledge. The subsequent chapters of this thesis unveil five novel systems that further build upon previous work to tackle these challenges.

Chapter 3

Unakite: Collecting and Organizing Online Information

Knowledge workers spend a significant portion of their time searching for solutions to their problems online. While numerous tools have been developed to support this exploratory process, in many cases, the answers to their questions involve trade-offs among multiple valid options and not just a single solution. In this chapter, we investigate this issue in the domain of programming and developers searching for solutions to their programming problems online. Through formative studies, we discovered that developers express a desire for help with decision-making and understanding trade-offs, and many answers to Stack Overflow questions describe such trade-offs. These findings suggest that *tools designed to help a developer capture information and make decisions about trade-offs can provide crucial benefits for both the developers and others who want to understand their design rationale.* We further probe this hypothesis with a prototype system named Unakite that **collects, organizes, and keeps track of information about trade-offs and builds a comparison table while searching and browsing, which can be saved as a design rationale for later use**. Our evaluation results show that Unakite reduces the cost of capturing tradeoff-related information by 45%, and that the resulting comparison table speeds up a subsequent developer's ability to understand the trade-offs by about a factor of three.

This chapter is modified from the following two published papers: (1) Michael Xieyang Liu, Jane Hsieh, Nathan Hahn, Angelina Zhou, Emily Deng, Shaun Burley, Cynthia Taylor, Aniket Kittur, and Brad A. Myers. 2019. "Unakite: Scaffolding Developers' Decision-Making Using the Web." *In Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19). Association for Computing Machinery, New York, NY, USA, 67–80.* (2) Jane Hsieh, Michael Xieyang Liu, Brad A. Myers and Aniket Kittur, "An Exploratory Study of Web Foraging to Understand and Support Programming Decisions," *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2018, pp. 305-306*

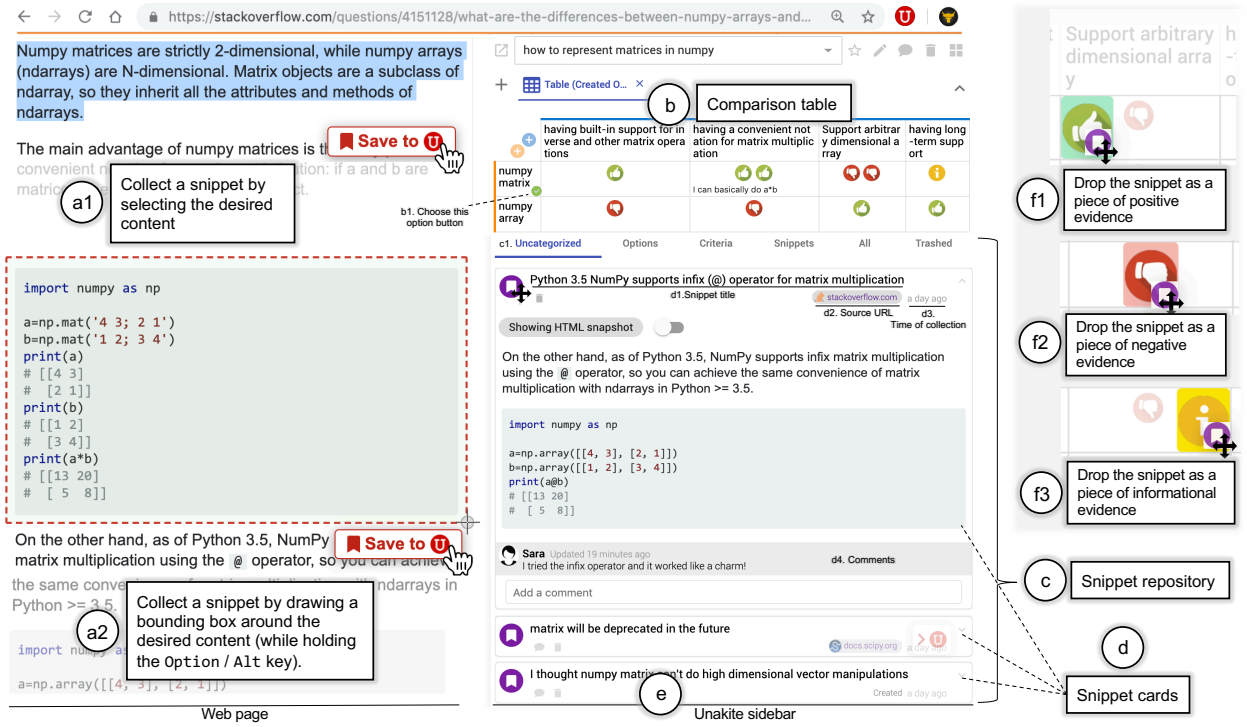


Figure 3.1: Unakite's user interfaces. With Unakite, a developer collects a snippet by selecting the desired content (a1) or by drawing a bounding box around the desired content (while holding the `Option` / `Alt` key) (a2) and clicking the "Save to U" button. The collected snippet immediately shows up under the "Uncategorized" tab in the snippet repository (c) as a snippet card (d) inside the Unakite sidebar (e), which shows the current task at the top ("how to represent matrices in numpy") along with the drop-down menu to pick other tasks and various tools for the task. The developer can quickly drag the snippet and drop it in one of the cells in the comparison table near the top (b). (f1-f3) show the details of the three parts of each cell in the table where the snippet can be dropped.

3.1 Introduction

Developers spend a significant portion of their time searching the web for answers [55, 310]. Past HCI and software engineering research supporting developers' foraging has focused on helping developers find a specific solution such as example code [54, 290, 291] and API documentation [310, 329], integrating it into one's own code [273, 358], and linking it back to the source [54, 273]. However, for many programming problems, there is no single correct solution – instead, there are many valid possible options (each with different trade-offs), and the decision comes down to how well each option matches the developer's goals [127, 209, 211, 281, 285, 295, 308]. For decision problems such as picking a JavaScript library to build websites, choosing an encryption algorithm to hash users' passwords, or seemingly straightforward ones like how to draw a blue circle on a web page, there is more than one good answer and trade-offs exist among all of the valid alternatives. For example, when picking a deep learning framework, Tensorflow [15] (with its top-notch performance and scalability) may be more suitable for building large commercial

AI systems, while a more approachable framework like PyTorch [14] may be a better choice for small academic projects and experiments.

As the number of frameworks, libraries, languages, and patterns increases [3, 5, 13], evidence about the trade-offs often needs to be collected across many competing information sources (e.g., documentation sites, blog posts, and discussion threads), and synthesized so that the developer can make an informed decision. Currently, this is a challenging process since there are high costs involved in capturing content, maintaining its provenance (its source), and synthesizing it with other content (that may very well be in different formats and structures) in a way that helps the developer to make a decision. For example, one developer in our formative study reported exactly these problems when copy-and-pasting relevant information into a Google Doc while deciding between using React [109] or Angular [134] to build her personal website.

This issue is compounded when later developers try to use the initial developers' code and discover that they need to understand why and how the original decision was made. Without proper documentation, it is hard for subsequent readers to figure out the context of the decision space: what options were considered, what criteria or constraints should be met, what the resulting trade-offs are, and what was deemed to be the most important and why. Indeed, understanding such *design rationale* is cited as one of the hardest questions for developers to answer about unfamiliar code [207, 208, 325].

In needs-finding interviews with 15 developers, we found that they expressed a desire for help with decision making and understanding others' design rationale when presented with decision problems involving multiple trade-offs. Next, we analyzed Stack Overflow (SO) questions, which revealed that many answers on SO contain information describing such trade-offs. These findings indicate that there are potential benefits to tools that help developers capture information, make decisions, and save the context for future reference.

To investigate the validity of this hypothesis, we built a prototype system called Unakite as a plugin for the Chrome browser. Unakite reduces the costs of capturing and organizing information about trade-offs, and persists this information so that it can serve as the design rationale. To reduce the burden on developers, Unakite provides these capabilities *while the user is searching and browsing*. Unakite is named after a pink and green semi-precious stone, and stands for "**U**sers **N**eed **A**ccelerators for **K**nowledge for **I**mplementations in **T**echnology **E**nvironments". It enables developers to easily collect content from any web page into an information repository. The amassed information is organized in a tabular format (which we selected based on evidence from our formative studies) that crystallizes the trade-offs among various solutions *in situ*. The

resulting organizational structures are automatically preserved and can be shared to support collaboration, documentation, and integration with code through comments.

We evaluated how well Unakite can support participants in collecting and organizing information about trade-offs as well as in understanding such gathered content. Compared to using Google Docs to build and maintain a comparison table, Unakite reduces the overhead cost of capturing tradeoff-related information by 45%. Compared to just going through unstructured information on a set of web pages, participants using Unakite were able to understand trade-offs involved in previously-made decisions about three times faster.

This work makes the following research contributions:

- formative studies showing developers' needs for support with decision-making,
- Unakite, a novel system that reduces the costs of capturing and organizing online information and preserves the knowledge as design rationale, and
- an evaluation of Unakite through two controlled studies that offer insights into its usability, usefulness, and effectiveness.

3.2 Formative Studies and Design Goals

To gain deeper insights into the barriers developers face about trade-offs, we performed two formative studies.

3.2.1 Study1: Interview with Developers

First, we conducted a series of needs-finding interviews with developers to understand how they currently collect and manage information about trade-offs in programming.

3.2.1.1 Methodology

Participants were a convenience sample of 15 developers (11 male, 4 female) recruited through social media listings and mailing lists. To capture a variety of processes, we chose 5 professional software developers, 2 doctoral students, and 8 master students. While we do not claim that this sample is representative of all developers, the interviews were very informative and helped motivate the design of Unakite.

We began by asking how frequently participants made decisions about trade-offs when programming. We then explored how they manage these situations. We asked the participants to provide context by reviewing their browser histories and code bases to cue their recollections

while retrospectively describing recent projects or problems. We solicited their workflows, strategies, mental models, frustrations, and needs. Finally, we wrapped up with questions probing their experience with understanding programming decisions made by other developers.

3.2.1.2 Results

Making decisions about trade-offs is frequent in programming. Almost all programming tasks described by participants involved some level of decision-making that required them to choose among options. In fact, 13 out of 15 said that they were frequently swamped with exploring multiple possible options while trying to compare them based on various criteria, such as the trade-offs among optimization methods when training neural nets (e.g., *"stochastic gradient descent"*, *"augmented Lagrangian"*, etc.) (P9) and the balance between cost and performance when *picking cryptographic algorithms* to protect users' sensitive information (P13).

Participants' browsing patterns and mental models for capturing trade-offs evolve as they dig deeper into the decision space, with a common representation being a comparison table. When approaching decision-making problems like *picking a JavaScript framework to build a web application* (P10), developers generally expected to find a quick-fix style solution at the beginning of their searching process. At this stage, they tended to only curate a short list of solutions that fitted their initial constraints as they queued each in a different browser tab for later reference, without pondering much about the advantages and disadvantages of each. As they dug deeper into the decision space (sometimes voluntarily doing due diligence to investigate multiple options before committing to something permanent (P4, P7), and other times because the previous solution they tried failed), they started to discover new options, criteria, and trade-offs that they were unaware of before. This naturally prompted them to go back to their earlier findings and make comparisons. As reported, their mental models at this stage quickly evolved into a comparison table, with its entries being filled according to information about whether an option satisfied a particular criterion. These findings prompted us to further analyze the applicability of tabular formats in synthesizing the trade-offs in programming problems, which we discuss in the next section.

No matter how organized their tabular mental models might become in the end, most participants reported that their exploration was inherently non-linear and tangled – there was no set pattern that was followed to acquire all the relevant information they needed. For example, as they went through web pages, they discovered new evidence, which in turn drove them to search for or go back to a previous page to read in detail about another option or criterion that they previously missed. This back-and-forth sensemaking process becomes particularly challenging,

as evidence is often spread across different web pages on different browser tabs, each with different formats and structures. Additionally, participants often do not realize that there are various trade-offs between options until they dig deeper into the decision space, at which point they are already overloaded with information and lost in browser tabs, and it is hard for them to recall, search for, or go back to previously missed content to fill in the blanks in their mental table. These findings prompted us to offer various features in Unakite to help developers go back to previously visited content such as automatically keeping track of the source URL and the scroll position when collecting information.

Both making decisions and understanding them later are difficult and cognitively demanding, and developers expressed a strong desire for tool support. 8 out of 15 said they used general-purpose tools and methods like taking notes in Google Docs or using a web clipper (such as that provided by Evernote) and reported problems such as: a high cost associated with collecting content (P7: *"...copy-pasting is just too much work, and I lose all the styling; while Evernote clipper clips the entire page, which is equivalent to not saving anything at all [because] I'd have to re-find it later."*); maintaining provenance (P15: *"...whenever I save something, I always forget to also save the URL [of the source]."*); synthesizing the new with existing content (P9: *"Evernote dumps everything I clip into a list of notes. There's no way for me to organize them."*); and guiding their exploration processes (P1: *"... sometimes there's just so much [evidence to find] that I often don't have a clue about what I'm supposed to search next."*). Additionally, participants reported that another disadvantage of using Google Docs or other applications like Evernote is that they must switch to another browser tab or application to access and organize their collected information. Such frequent context switches are tedious and have been shown to harm developers' productivity [129, 185, 256]. These findings inspired us to help developers easily externalize their mental models when they are searching and browsing, by providing an easier method of tracking and deciding among available options.

Almost half (7/15) of the participants admitted that they do not document their decisions anywhere. An additional three said that they would only record important source URLs in code comments. Interestingly, participants also discussed the difficulties in code comprehension, particularly when trying to understand code written by others that involved unexpected decisions. They attributed the frustrations primarily to being unable to uncover the context of the decisions and the original trade-offs, and fearing they might accidentally violate important yet hidden constraints that guided the original decision, which is congruent with prior research [197, 206]. This motivated Unakite to automatically keep track of the initial developer's decision making trails

as the design rationale, unlike prior work where developers are forced to manually create documentation of decisions after they are made [280, 344].

3.2.2 Study 2: Analysis of Stack Overflow

Stack Overflow (SO) is an important tool for answering programming questions, and participants cited it as their most frequently visited resource. Given this motivation, we undertook an analysis to assess the proportion of questions on SO which capture trade-offs among multiple options and to determine if the tabular format identified in the interviews is indeed an appropriate structure for synthesizing these trade-offs.

We utilized two sets of posts for this analysis. First, we queried the 50 most viewed questions. We were concerned about this sampling method as it may only represent a narrow set of topics which happen to be the most popular, whereas the average developer may have more niche interests [45]. To obtain a sample of questions with a variety of topics that may be more representative of the interests of the general population, we collected another 90 questions by querying for posts created on a particular day which contained three or more answers. Through manual analysis and construction of comparison tables using spreadsheets, we found that the trade-offs contained in 88% of the 50 most-viewed and 49% of the 90 general population questions along with their answers could be reasonably organized into tables. In fact, we found that some answers already included tables to summarize the trade-offs among the options, e.g., [8, 9]. Together with the results from the interviews, these findings motivated the design of Unakite's organization features that let users synthesize information about trade-offs into comparison tables.

3.2.3 Summary of Design Goals

Led by our formative findings and prior research discussed in chapter 2, we hypothesize that an effective interface for decision making about trade-offs while sensemaking should support:

- **Scaffolding:** helping developers form systematic models when approaching decision making problems with trade-offs.
- **Lightweight interactions:** reducing the cost of collecting and organizing content so that the entry barriers for developers to use the tool are low.
- **Summarization:** helping developers synthesize and summarize different pieces of content together and manage them, as suggested by prior work [262, 378, 379].
- **Contextualization:** enabling developers to recreate the context from which information snippets were collected and copied for better sensemaking [283, 307, 321].

3.3 Unakite

Guided by the design goals above, Unakite enables developers (both experienced and novice) to easily collect any content from any web page into *snippets* (pieces of information) and organize them by *options*, *criteria*, and *evidence* as they are searching and browsing the web, and thereby keep track of their decision-making trails for later reference. Unakite is an extension to the Chrome Web browser and a web application.

We first illustrate the experience of using Unakite by describing an example usage scenario that embodies many of the use cases identified in our formative studies.

3.3.1 Example Usage Scenario

Sara, a junior professional developer, is tasked with writing Python code to handle matrix calculations for her company. As the code will be used in production, she wants to determine the best way to represent matrices using `numpy` [10] before starting the implementation. She decides to use Unakite to help her stay organized during her exploration process.

Sara logs into Unakite, enables it on her current web pages, brings out the Unakite sidebar (Figure 3.1-e), and selects "Create a new task", entering "how to represent matrices in `numpy`" as the task name. Next, she starts a Google search on this topic.

As she goes through the search results, she comes across an SO page about the differences between `numpy matrix` and `numpy array`. She then quickly collects text describing both numpy matrix and numpy array into the task snippet repository by just selecting the text and click the "Save to U" button that pops up (Figure 3.1-a1). The collected snippets immediately appear under the "Uncategorized" tab (Figure 3.1-c).

Continuing on, she comes across several criteria that seem to be good standards to evaluate which of the two options she just discovered is better. For example, she thinks that "having a convenient notation for matrix multiplication like a*b" is essential for the readability of the code. Therefore, Sara collects those criteria using the same mechanism.

As the number of collected snippets gets larger, Sara decides to quickly organize them by simply dragging and dropping each snippet into the comparison table (automatically created along with the task) above the snippet repository in the sidebar (Figure 3.1-b). For example, she drags `numpy matrix` into one of the row headers as an option (e.g., a possible solution to solve the task). After a basic table structure is laid out, she realizes that an optimal method should not be deprecated in the future, so she clicks the blue "plus" button to create a new column and types

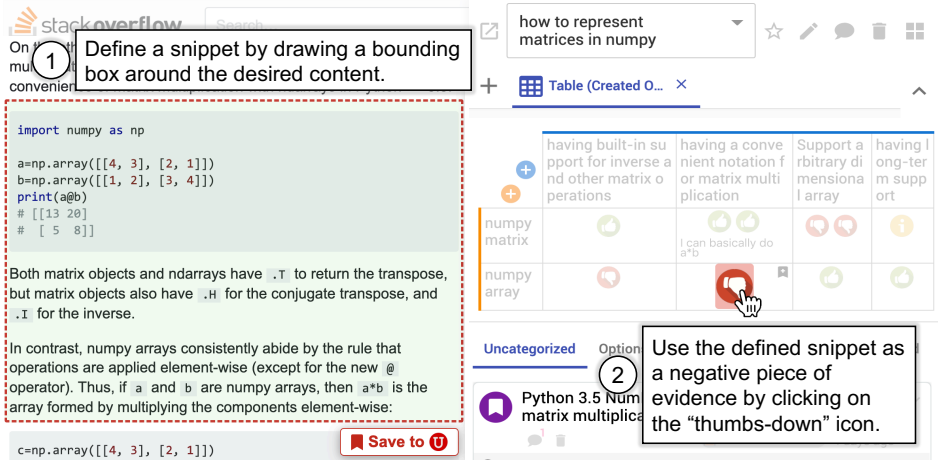


Figure 3.2: "Teleporting" content directly into the comparison table as a piece of evidence.

in "having long-term support" as a new criterion. As it's not one of her immediate concerns, she drags that column to be the last one in the table.

To save a section of the SO page that compares the two options in terms of the criteria she just collected, Sara uses the *snapshot* feature (holding the `Option` / `Alt` key and using the mouse to drag on screen) to draw a bounding box around that section (Figure 3.1-a2). Instead of clicking the "Save to U" button to save it as a snippet and then drag it into the table (which she certainly can), Sara uses the *teleport* feature (Figure 3.2) by clicking on one of the rating icons in the corresponding table cells to directly save the snapshot as a snippet and use it as a piece of evidence. For example, she gives `numpy matrix` a "thumbs-up" (positive rating) for "having a convenient notation for matrix multiplication like a*b" and `numpy array` a "thumbs-down" (negative rating) for "having built-in support for inverse and other matrix operations". Alternatively, developers could also label a snippet as "informational" if it does not have a positive or negative effect on their decision (Figure 3.1-f1,f2,f3).

After filling up the table with options, criteria, and ratings (evidence), Sara now feels clear that `numpy matrix` should be the better choice, so she clicks the green "Choose this option" button (Figure 3.1-b1) next to that option to indicate it was chosen. She wants to document her decision in the company's internal documentation site. The table she organized, along with all the information snippets she collected, is automatically preserved by Unakite for the current task. She clicks the "Open task detail page" button to open the task in the Unakite dashboard web app, copies the URL from the address bar, and pastes it into her code documentation with "Here's how I decided to choose `numpy matrix`".

A year later, Larry comes in and reads the code along with the Unakite table that Sara created. He glances the ratings and checks the evidence snippets by mousing over the rating icons. He

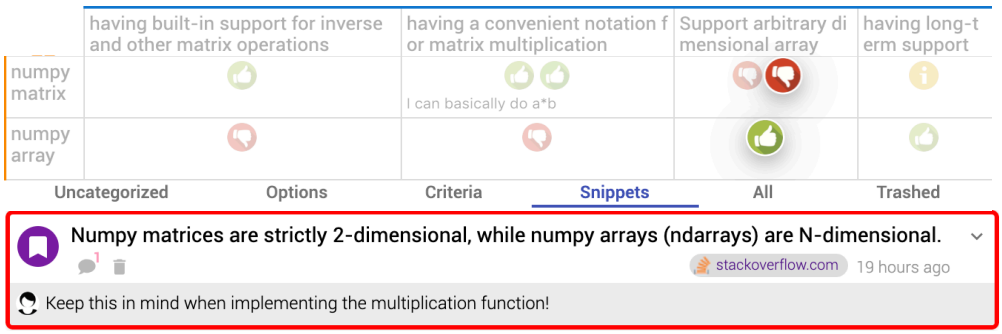


Figure 3.3: A snippet used as evidence in multiple cells. Selecting a snippet will highlight its location(s) in the table.

quickly understands Sara's decision, and realizes the opportunity to switch to using a `numpy array` since now the code needs to be able to perform vector operations in arbitrary dimensions and be supported in the long term, both of which are criteria that Sara identified previously.

3.3.2 Detailed Design

3.3.2.1 Scaffolding

Unakite provides developers with scaffolding when managing decision making tasks that involve trade-offs by offering the "Option-Criterion-Evidence" (OCE) framework as illustrated in the example scenario. A user can create as many tasks as desired, where typically each task represents a different decision. For each task, the information is organized in a tabular format (Figure 3.1-b) where options are the row headers, criteria are the column headers, and pieces of evidence are spread across the rest of the cells.

We provide this framework for several reasons. As mentioned in the interview study results, developers' mental model for capturing trade-offs is similar, but less organized, to that described in this framework. Formalizing it provides a concrete framing for developers to think about decisions in a structured way that they are already familiar with. Another aim of providing this structured framework is to encourage developers to think about trade-offs from the start to avoid the unnecessary frustrations later on (as described in the interview results).

3.3.2.2 Lightweight Interactions

Unakite offers various lightweight interactions to collect information and organize them according to the OCE framework. It provides two intuitive ways to collect any content from any web page. The first is selecting the desired content using the cursor in the normal way, and then clicking the "Save to U" button that pops up (Figure 3.1-a1). Another way to collect large pieces of

information (code snippets that span multiple lines, columns or sub-sections of tables, pictures, etc.) is to use the snapshot feature: drawing a bounding box around the desired content (Figure 3.1-a2 and Figure 3.2) and clicking the "Save to U" button. These interactions are carefully designed based on developers' natural habits of copying-and-pasting content and links and taking screenshots without introducing an extra cognitive load of learning a new interaction, and thereby reducing the starting cost for developers to use Unakite.

Unlike previous tools where information was saved either in pure text format [192, 193] or as raw HTML without CSS styling [364], Unakite combines the best of both copying-and-pasting and taking screenshots by capturing, saving and later showing the content of a snippet with its original styling and including the rich, interactive multimedia objects supported by HTML, like images and links. This feature makes the content in snippets more understandable and useful, and also helps developers quickly recognize a particular snippet among many others in the repository by its appearance. Typically, developers will include example code in the snippets as copied from SO and other sources, and Unakite is careful to preserve the formatting of the code, so it can later be copy-and-pasted into the user's code once a decision to use it has been made.

The collected snippets will be displayed in the current tasks' snippet repository (Figure 3.1-c), which serves as a container that holds all the collected snippets in the form of snippet cards (Figure 3.1-d). One of the benefits of having this repository is that it serves as an information buffer between the web and the comparison table: as recommended by Kittur et al. [193], a "two-stage" model in which information is first saved and then organized, results in a better "structured information space".

To solve the problem of frequent context switches (identified in the interview study), Unakite brings the ability to access and organize collected information directly into the browser tab that the developer is currently using – Unakite provides a sidebar (inspired by [347, 348]) on the right side of the current window (Figure 3.1-e) containing the comparison table (Figure 3.1-b) and the aforementioned snippet repository. There are several major advantages for developers using the Unakite sidebar. It serves as a comprehensive dashboard that contains both the collected information and the ability to organize them into comparison tables (discussed later in detail) all in a small footprint. Unlike PlayByPlay [364] in which the sidebar lives in a part of the browser UI, Unakite's sidebar is directly injected into the DOM tree and therefore can provide rich interactions with the original web page. The sidebar can be toggled in and out like a drawer using the keyboard shortcut `Ctrl +` ` (backtick) or using the "Open/Close Unakite Sidebar" button on the bottom right of the window. When it opens, it automatically shrinks the width of the web page body to make sure nothing is visually hidden.

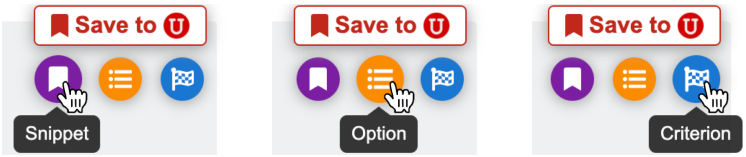


Figure 3.4: Mousing over the "Save to U" clip button will reveal three additional buttons to collect the desired content specifically as a snippet, an option, or a criterion.

Unakite provides easy and intuitive interactions such as drag-and-drop, allowing users a variety of ways to quickly organize the collected information into a comparison table. A developer can drag a snippet card from the snippet repository and drop it into the table as either a row header (so it is an option), a column header (as a criterion), or into a cell as a piece of evidence, just as Sara did. Inspired by prior work [262], one can "rate" a snippet as either a positive (shown as a "thumbs-up" rating icon, see Figure 3.1-f1), negative (shown as a "thumbs-down" rating icon, see Figure 3.1-f2), or informational (shown as an "info" rating icon, see Figure 3.1-f3) piece of evidence. Moreover, a snippet can be reused as the evidence in multiple cells. Selecting a snippet (by clicking on it, see Figure 3.3) in the snippet repository will reveal its location(s) in the comparison table, and selecting an icon in the table opens the corresponding snippet in the repository.

There are two additional shortcuts to put snippets directly into a table. To collect some content as an option or a criterion, one can mouse over the "Save to U" button and click the "Option" or the "Criterion" button (Figure 3.4) that appears below. This is modeled after the various options for "liking" in Facebook. In addition to collecting the desired content as a snippet, this will automatically create a new row or column in the comparison table. Another shortcut is the teleport feature that Sara used above (Figure 3.2). These shortcuts are enabled by and add additional benefits to Unakite's always-available sidebar. Together with the other features described above, users have the flexibility to capture and organize their knowledge in various ways and in any order without needing to follow a preset process.

As illustrated in the example scenario, every Unakite task, including all of its snippets and comparison tables, can be accessed in the Unakite web app via a unique URL in any browser. This makes sharing and keeping track of one's decision easier and more powerful: developers can choose to share the link to a task via email to their friends and colleagues to show how and why the decision was made, and the link can be embedded in documentation or comments in code, preserving the actual trade-offs and design rationale in addition to where any example code was copied from.

3.3.2.3 Summarization

Unakite introduces several levels of summarization to help developers manage and digest information.

The comparison table provides a high-level summary of the decision making space and the trade-offs among various options. It offers a clear and glanceable picture of the advantages and disadvantages of each option through the "thumbs-up" and "thumbs-down" rating icons without having to expose the nitty-gritty details of the evidence content, which is useful both for the developer making the decision and later code readers, as shown in the example scenario. Additionally, it serves as a presentation of one's exploration progress that helps users understand which part of the decision space has been explored and which has not (revealed in the interview studies as an important clue developers need when exploring multiple options). For example, the empty cells in the table provide developers with clues about where they need to focus next.

The individual rating icons provide another level of summary of their corresponding supporting evidence. Unlike in previous summarization tools [379] where contents are recursively summarized into words, Unakite encourages the user to parse out the information in a snippet that captures the relationship between an option and a criterion, and represent them as rating icons. We believe this mechanism can usually capture developers' information needs of whether an option satisfies a specific criterion, as identified in the formative interviews. One can also manually add a rating leveraging their prior knowledge directly in the table by clicking the "Add a snippet" button on the top right of the table cells, and just type or paste. To dig into the detailed evidence of each rating, users can simply click on those icons in the sidebar tables or mouse over the icons in the Unakite web app to reveal the supporting snippet card.

In addition to the built-in summarization mechanisms above, Unakite also enables users to note down their own summaries in various places. Users can easily edit the snippet title (Figure 3.1-d1) in the snippet card to be something more summative. For example, for a long snippet that talks about the performance advantages of React [109] over Angular [134], a user may summarize it as "React apps load faster than Angular ones." There is also a text box in each table cell for users to summarize all the evidence in that cell or keep track of the evidence that cannot easily be captured by rating icons, such as prices and speed. Moreover, one can add comments to snippets (Figure 3.1-d4), table cells, and tasks about their opinions, thoughts, or the results of their experiments with an option, etc. These were added based on feedback that developers needed more flexibility to add comments and content in many places.

3.3.2.4 Contextualization

Meta information such as the URL of the source web page (Figure 3.1-d2) and the time of collection (Figure 3.1-d3) are automatically recorded along with the snippet and displayed on the snippet card in Unakite. Using this feature, developers are able to go back to the web page where a snippet was collected. Unakite will even help developers to go back to the exact scroll position where the snippet was collected if possible, saving the extra effort of locating it on a web page. The time when a snippet was collected is especially useful in giving developers a rough estimate of the age of the information and helping them determine whether it is still valid (e.g., API methods might be deprecated or trade-offs might change in newer library versions).

3.4 Evaluation

We conducted two initial user studies of the Unakite system in order to answer the following questions:

- Can developers collect and organize information using Unakite?
- How does Unakite compare to currently available tools like Google Docs?
- Do Unakite tables offer value over just reading through web pages when trying to understand the design rationale?
- How can the design of Unakite be improved?

3.4.1 Study 1 - Authoring Unakite Tables

We carried out a study to evaluate developers' ability to use Unakite to collect and organize information about trade-offs.

3.4.1.1 Procedure

We recruited 20 participants (15 male, 5 female) aged 23-37 (μ = 26.75, σ = 3.49) from a local research participation pool. The participants were required to be 18 or older, to be fluent in English, and to be experienced in programming. Participants had on average 8.8 years of programming experience, with the longest being around 15 years. 13 participants had professional programming experience, with the rest having experience in college.

In this study, participants were first presented with two tasks each: *(A) how to invoke a function in JavaScript* and *(B) how to create or update a resource using REST APIs.* For each task, they started from scratch without using any information snippets from previous tasks. The study was

	# manually created snippets / # snippets	# options	# criteria	# ratings	# pos. ratings	# neg. ratings	# info. ratings
Task 1	0.70 (1.34) / 12.10 (3.38)	2.30 (0.67)	2.70 (1.57)	8.80 (4.10)	3.00 (1.89)	1.80 (2.30)	4.00 (3.80)
Task 2	1.20 (3.16) / 17.50 (4.48)	2.60 (0.52)	4.60 (2.07)	13.20 (4.42)	7.70 (4.08)	2.60 (2.46)	2.90 (2.42)
Task 3	2.00 (3.77) / 18.89 (8.31)	3.74 (1.37)	4.74 (2.58)	12.58 (8.87)	6.37 (5.24)	3.84 (4.29)	2.37 (2.52)

Table 3.1: Statistics for various Unakite feature usages in Study 1. Statistics are presented in the form of **mean (standard deviation)** in the table.

a between-subjects design, where participants were randomly assigned to either the Unakite or the control condition. In the Unakite condition, participants were given a static web page adapted from a real Stack Overflow page discussing the task topic in each task. Participants were asked to use Unakite to collect and organize information from that single page into a comparison table, and were instructed to inform the researcher when they thought they had finished the task or felt like they could make no further progress. In the control condition, participants were asked to do the same but to build comparison tables using Google Docs instead. We deemed Google Docs as a proper baseline since: 1) it was reported in the formative study as a common tool people use to take notes while making decisions; 2) all participants in this user study were already proficient in using it; 3) compared to other solutions like spreadsheets, it can be easily used to capture richer contexts such as formatted text (example code), images (screenshots of execution results), and links (URLs of documentation and tutorial pages).

All participants were then given a third task in which they were asked to use Unakite to help them understand the trade-offs and make decisions on whatever programming problems they were trying to solve in real life.

Participants in the Unakite condition were given a 10-minute tutorial showcasing the various features of Unakite and a 5-minute practice session before starting. Those in the control condition were given the same tutorial and practice session before the third task. At the end of the study, the researcher conducted a survey and an interview eliciting subjective feedback on the Unakite experience. In particular, participants were asked to list 3 of their favorite features as well as 3 least favorite features or possible improvements of Unakite. The study took about 80 minutes per participant, using a designated MacBook Pro computer with Chrome and Unakite installed. All tasks were screen-recorded for later analysis. All participants were compensated $20 for their time.

3.4.1.2 Results

All participants were able to complete all of the tasks in both conditions. As shown by the statistics in Table 3.1, the Unakite participants were able to use the various features to collect and organize information into comparison tables.

To examine how Unakite performs compared to the control condition, we opted to compare the *overhead cost* of using both tools to collect and organize information. For the Unakite condition, the overhead cost is defined as the portion of the time participants spent on directly using Unakite features (selecting, snapshotting, dragging snippets into the comparison table, etc.) out of the total time they used for a task, since the rest of the time was spent reading and understanding the Stack Overflow page. Similarly, for the control condition, the overhead cost was calculated as the percent of time participants spent on copy-and-pasting content, making screenshots, and staying on the Google Docs browser tab to organize the table.

We conducted a mixed-effect linear regression with overhead cost as the outcome, condition, task, and their interaction as fixed effects. Since participants may have different abilities in performing the tasks, we included a random intercept for each participant. Results show that the overhead cost when using Unakite is significantly lower (coefficient $= -0.22$, $t(18) = -4.81$, $p = 0.0001$) than the control condition, while task (coefficient $= -0.05$, $t(18) = -1.40$, $p = 0.1777$) and the interaction term (coefficient $= 0.04$, $t(18) = 0.71$, $p = 0.4861$) does not have an effect on the overhead cost. Across both tasks, the average overhead cost was reduced by 45% when using Unakite (Mean overhead cost $= 25\%$, SD $= 0.07$) compared to using Google Docs (Mean $= 44\%$, SD $= 0.12$). Thus, using Google Docs did add a lot of extra time, whereas using Unakite, even though unfamiliar, was quick and non-disruptive.

In the survey, participants reported (in 7-point Likert scales) that they thought the interactions with Unakite were understandable and clear (Mean = 6.20, Median = 6.00, 95% CIs = [5.84, 6.56]), they enjoyed Unakite's features (Mean = 6.00, Median = 6.00, 95% CIs = [5.52, 6.48]), and would recommend Unakite to friends and colleagues doing programming work (Mean = 6.20, Median = 6.50, 95% CIs = [5.75, 6.65]).

Nine of the 20 participants requested that we send them the URL of their third task that they created using Unakite for reference and five of them asked us to help them install Unakite on their computer for personal use and future updates, highlighting both the utility of the system as well as the realism of the tasks they chose. Figure 3.5 shows P13's table capturing the trade-offs in choosing JavaScript front-end frameworks.

Another highlight in the study is that P3, P10, and P18 decided to either commit or switch to the option they identified as the best option after using Unakite to build comparison tables on the topic of their choosing. For example, P3 researched on hybrid AR development frameworks that can take advantage of both ARCore [11] on Android and ARKit [12] on iOS, and found ViroReact [16] to be the best choice. A quick follow-up interview a week later revealed that he had already begun using that framework, and it did satisfy all of his needs so far.

	Availability of Learning Resources	Popularity	Ease of Integration (with Other Libraries)	Core Features	Usability
React	👍	👍	ℹ	ℹ	👍 👎
Angular	👍	👍	👍 👎	👍 👎	👍
Vue		👎	ℹ	ℹ	👍 👎
EmberJS	👍 👎	👎	ℹ	👍 👎	👍 👎

Figure 3.5: Participant P13's comparison table capturing the trade-offs in choosing JavaScript front-end frameworks.

3.4.2 Study 2 - Understanding Unakite Tables

We carried out a second study to evaluate whether developers could understand the trade-offs encapsulated in comparison tables and snippets previously built by others using Unakite.

3.4.2.1 Procedure

We recruited 16 participants (9 male, 7 female) aged 21-32 (μ = 25.3, σ = 3.19) from the same local participation pool as in Study 1 (but no-one participated in both studies). Participants had on average 7.8 years of programming experience, with the longest being 17 years. None of them were familiar with either the topics involved in this study or Unakite. The study took about 40 minutes per participant, using the same setup as in Study 1. All participants were compensated $15.

Participants were given a 10-minute tutorial showcasing the various features of the Unakite web app. The study was a within-subjects design, where the participants were presented with two tasks of roughly equal difficulty and were asked to solve one of them with the help of Unakite and the other by reading through a set of web pages, in a counterbalanced order. For each task, participants were given some code written by the researcher to solve a problem, some necessary background information about the problem, and a list of options that were available to solve it. They were then asked to explain why the decision was made to choose the particular option used in the code and the associated trade-offs. In the experimental condition, participants were provided with a previously-built structure (including the comparison table and the snippet repository) through the Unakite web app, while in the control condition, participants were instructed to only read through the set of web pages that the structure in the experimental condition was built from. Specifically, the two tasks were to explain the decision and the trade-offs of:

- Choosing `numpy array with Python 3.5+` instead of `numpy matrix` or `numpy array with Python 2.7` to perform some matrix calculations like multiplication, inversion, element-wise multiplication, etc.
- Choosing `numpy array` instead of `Python list` or `Python array` to hold data involved in large-scale numerical manipulations such as regression analysis.

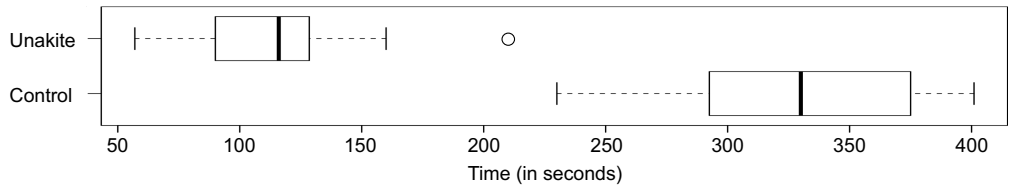


Figure 3.6: Box plot of the average task completion time for the participants under different conditions: Unakite vs. Control in Study 2.

To ensure realism, both tasks were based on actual questions asked and answered on Stack Overflow that are heatedly discussed and well-maintained by real developers.

3.4.2.2 Results

Two researchers each listed all possible explanations to the two tasks independently. After resolving conflicts, we produced a list of possible explanations for each task as the gold standard. To quantitatively evaluate participants' performance, we measured the time it took for them to offer three legitimate explanations - those within the gold standard list - in each condition, which all participants were able to accomplish.

A two-way repeated measures ANOVA was conducted to examine the within-subject effects of condition (Unakite vs. Control) and task (A vs. B) on task completion time. There was a statistically significant effect of condition ($F(1,26) = 25.59$, $p < .001$) such that participants completed tasks significantly faster (almost 3 times faster) with Unakite (Mean = 114.63s, SD = 38.91s) than in the control condition (Mean = 332.56s, SD = 56.26s), as visualized in Figure 3.6. There was no significant effect of task ($F(1,26) = 0.01$, $p = 0.94$), indicating the two tasks were indeed of roughly equal difficulty.

3.4.3 Evaluation Discussion

3.4.3.1 Usability and Usefulness of Unakite's features

The snippet collection features, including both the selecting and the snapshot features, were considered highly useful, with 15 participants citing them as one of their favorite features. Participants said they were *"the perfect combination of copy-pasting and taking screenshots"* (P15) with the additional benefits of *"retaining the original styling [of the collected content], especially when there's code"* (P9), *"keeping track of the [source] URL"* (P7), and *"saving [users] some typing"* (P5). The drag-and-drop interactions were also popular, receiving 13 mentions in participants' "top three" lists, primarily due to its ease of use (P18: *"it is natural, like picking things up and dropping them in buckets"*). Participants also appreciated that the design of the Unakite UI is clean and easy

to learn (12/20), and the overall experience was satisfying (10/20). The sharing via URL feature also received nine mentions, with participants laying out potential usage scenarios like *"putting it in code comments or [their lab's] internal documentions"* (P11), *"using it for presentations in code reviews"* (P8), *"attaching it in emails that explain my code"* (P5), etc.

Compared with using Google Docs, P15 praised the value of Unakite's snippet repository functioning as an information buffer: *"It's like a note-taking space. I can just easily grab as much info that's related to my topic as I want, and they don't have to directly fit into the table, but can be something interesting to use later on; whereas in Google Docs, the cost of buffering these interesting snippets somewhere is pretty high."*

Participants have mixed opinions on how summarization works in Unakite. Most of them (16/20) agreed that summarizing snippets into positive, negative, or informational icons alleviates their burden of having to manually look at the content of each snippet every time, and makes the comparison tables much more skimmable, e.g., *"visual interpretation of thumbs ups and downs provides a quick summary"* (P18). However, P17 also pointed out that *"value comparisons between criteria (columns) are difficult,"* suggesting some notion of weight should be applied differently to the columns when construing the table. P3 indicated that the meaning for the thumbs-up/down icons is open for interpretation in a sense that *"having more thumbs-ups does not necessarily mean [that an option] is better [in terms of a criterion], it could simply mean that the author found more positive evidence, unless she specifies that [more means better] in the first place."* Based on these valuable insights, we believe that there are new interface design opportunities for future work to explore so that the value of Unakite-style comparison tables could be further improved.

3.4.3.2 Usage Patterns

Similar to what Morris et al. found [262], there was an unbalanced use of the positive and negative ratings in the study: positives (228 in total) are more heavily used than negatives (117 in total). A possible explanation for this asymmetry is that people in general lean towards finding and keeping track of evidence of what "works" rather than what "doesn't work".

Participants exhibited two major usage patterns when interacting with Unakite: (1) collecting-oriented: alternating between *long* collecting stages (in which they keep collecting content into the snippet repository) and *short* organizing stages (in which they focus on putting the collected snippets into the comparison table); or (2) organizing-oriented: all snippets going directly into the comparison table immediately after they are collected. We are delighted that interactions in Unakite are flexible enough to support both usage patterns equally well.

The studies showed some evidence that Unakite might also be used for other tasks like comparison shopping for electronics or makeup, even though they are not the focus of Unakite.

3.5 Discussion and Future Work

Through designing and evaluating Unakite, we gained deeper insights into people's frustrations and needs towards making sense of programming trade-offs on the web. This could pave a path for future work.

To support cases in which the needs for collecting and organizing information are not discovered until partway through an investigation process, future research can explore automatically summarizing exploration paths in the background so that developers can retroactively organize their work with reduced overhead. This is, in fact, explored to some extent in our Crystalline system discussed in chapter 4.

One can also investigate the use of Unakite as a pedagogical tool. Many areas of computer science (e.g., data structures, systems) require students to consider different options in terms of trade-offs, rather than determining a single correct answer. Anecdotally, many students find this difficult. The exercise of creating a comparison table to explicitly compare multiple options for a task (e.g., using a stack or a queue to build an undo function) would force students to explicitly determine the criteria necessary for the task, gather evidence to support ratings, and make an educated decision based on these ratings.

Several participants mentioned in the interviews that Unakite's is *"useful in terms of helping [them] form mental models"* (P4) while searching, especially when there are a lot of equally plausible choices involved. However, P15 also pointed out that the table structure is *" a double-edged sword"* in a sense that it promotes structured thinking but also *"forces [users] to follow a fixed pattern."* In light of these mixed opinions, future work could conduct a long-term field study with two specific goals in mind: (1) exploring the possibility of making the current version of Unakite an intervention mechanism to promote a structured way of approaching decisions about trade-offs and help developers form the habit of staying organized; and (2) exploring different schema of knowledge representation other than tables such as decision trees that could also support developers' decision making about trade-offs and beyond.

Chapter 4

Crystalline: Automating Information Collection and Organization

Our previous system Unakite encouraged authors to document their decision-making processes and results using the tool's lightweight collecting and organizing features. However, it remains a laborious process for people to manually identify and clip content, maintaining its provenance and synthesizing it with other content. In this chapter, we explore the idea of having a system dynamically help users keep track of and organize information by leveraging the content they are browsing and the signals from their browsing behavior. We instantiate this idea in a prototype system called Crystalline, which plays the role of a user's copilot and attempts to automatically identify and keep track of the options, criteria, and corresponding evidence snippets from the web pages that a user has viewed, and organize the snippets into both list and tabular formats with prioritization. The goal is that users can focus on reading and understanding web content while occasionally guiding the system when it makes mistakes. Our lab study suggests that users are able to create comparison tables about 20% faster with a 60% reduction in operational cost (compared to using Unakite) without sacrificing the quality of the tables.

4.1 Introduction

Developers spend a large portion of their time searching and making sense of the web for solutions to their programming problems [55, 310]. In many cases, the answers to such problems

This chapter is modified from the following published paper: Michael Xieyang Liu, Aniket Kittur, and Brad A. Myers. 2022. "Crystalline: Lowering the Cost for Developers to Collect and Organize Information for Decision Making." *In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 68, 1–16.*

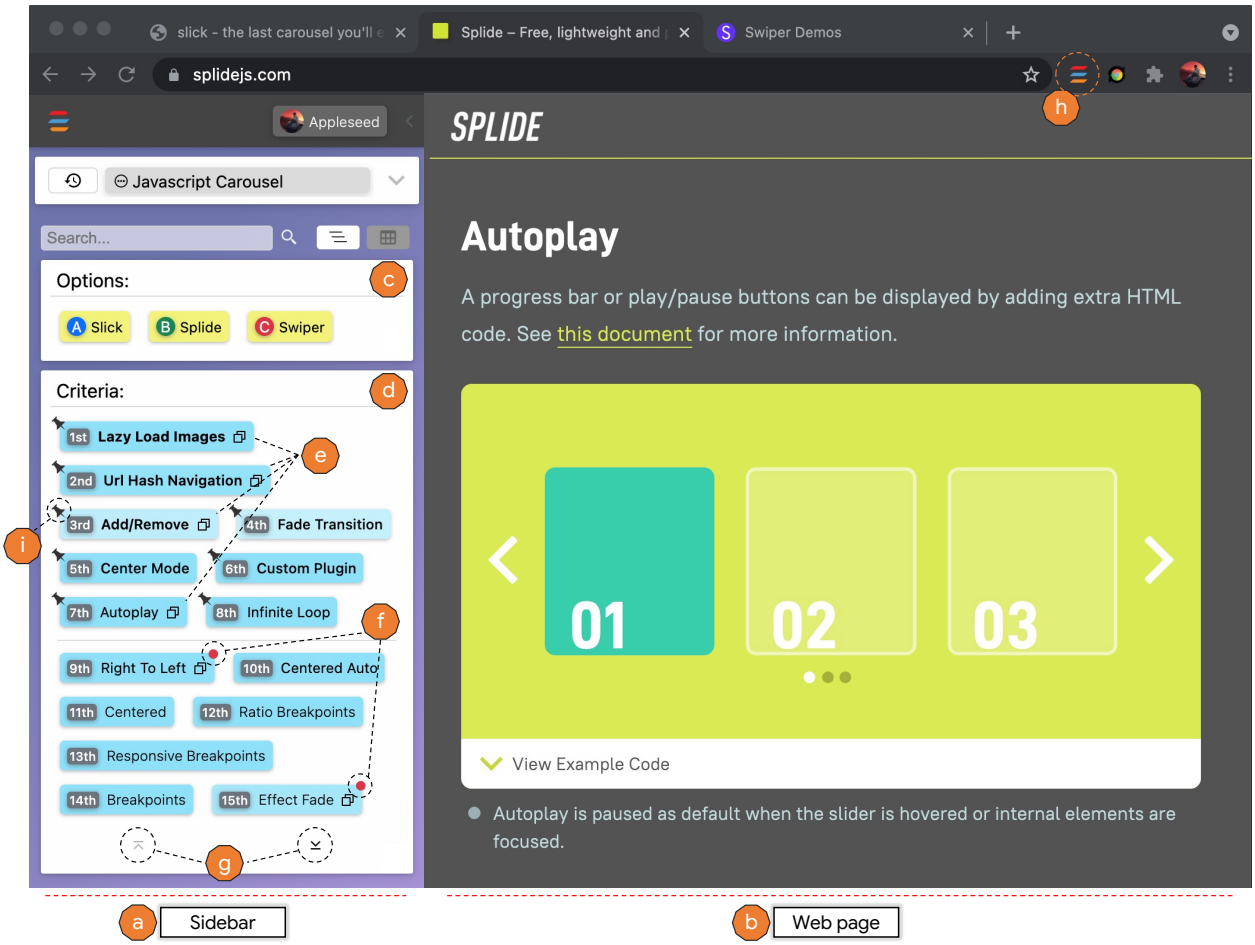


Figure 4.1: Crystalline's list view UI (a). As the developer browses a web page (b), Crystalline attempts to automatically collect options and criteria from the page, and display them in the options (c) and criteria panes (d) in the sidebar (a). In addition, Crystalline leverages natural language processing to automatically group similar criteria together, as shown by the multiple-pages icon (e). Crystalline uses behavioral signals such as mouse movement and dwell time to try to automatically detect the relative importance of the criteria (shown by the display order, with most important at the top). Users can use the "See more" and "See less" buttons (g) to adjust how many criteria are to be displayed at once. Crystalline will remind users of the existence of additional related evidence through a red notification dot at the top right of a criterion (f). The sidebar can be toggled in and out by clicking the browser extension icon (h). Users may pin (i) important criteria to the top of the list.

are not limited to a single solution, but developers discover that there are multiple legitimate options, and they must identify relevant criteria and constraints based on their unique contexts and carefully consider the trade-offs among those possible options [127, 166, 209, 211, 232, 233, 281, 285, 295, 308]. For example, when converting an old web application to use a modern JavaScript front-end framework, React.js [109] (with its ability to be progressively adopted into existing code bases) may be more suitable when one wants to gradually convert each separate module while minimizing the overall system downtime, whereas a more comprehensive framework such

as Angular [134] might be a better choice if one wants to take advantage of various official utility packages like routing [132], animation [131] and data validation [130].

There have been many commercial and research tools and systems that try to help people make sense of information about trade-offs to facilitate further decision-making, such as by helping with easily capturing snippets of information [46, 133, 145, 315, 348] from web pages or organizing and synthesizing information into useful schema and representations [73, 101, 163, 192, 232, 354]. For example, one common practice that people employ is copying pieces of text as well as taking screenshots and putting them in a running Google Doc as they search and browse the web [277]. One system that is relevant to the context of programming is Unakite [232] (chapter 3), which enables developers to collect and organize information online into comparison tables with options, criteria, and evidence to help with making decisions (see Figure 3.1).

However, even with the above tools, it remains a challenging process for developers to *manually* identify and capture the relevant content, maintain its provenance (where it came from), and synthesize it with other content. Prior work suggests that one cause is that people are often uncertain about which information will eventually turn out to be relevant, valuable, and worth capturing, especially at early stages of their learning and exploration when they are overloaded with information [41, 114]. Under these circumstances, people are hesitant to frequently pause and shift their focus from the investigation itself to reasoning about what to capture for later use [72, 159, 193, 312], or they could be too engaged in the sensemaking process and forget to collect anything at all. Indeed, research suggests that interactions for gathering information while performing active reading need to be quick and low effort, otherwise people tend not to capture information in the first place [159, 232, 247, 337]. In addition, though existing tools provide users with the flexibility and agency to synthesize the collected information into useful representations, such as comparison tables [73, 232] or knowledge maps [269], developers still need to perform these organizing operations manually. This is often a laborious process, as developers need to take stock of all the pieces of information, identify connections among them, and directly manipulate the representation to reflect the connections.

Another challenge reported in prior work is that developers' needs for collecting and organizing information are often not discovered until part of the way through an investigation process [75, 232]. This could be due to several major reasons, including but not limited to: 1) additional external requirements, constraints, or user feedback are discovered or introduced in the middle of a project which significantly complicates the original decision making problem [86, 103, 104]; 2) developers discover many more options, criteria, and their trade-offs than they anticipated at the beginning [232]; and/or 3) developers are required to explain or document

their decisions and design rationale after the fact for the long-term maintainability and success of a software project [94, 119, 207, 208, 212, 300, 325]. In these situations, it is hard and involves duplicate work for developers to recall and retrace their steps for reaching their current state of sensemaking (the linear history visualization in almost all current browsers is known to be not particularly effective [75, 179, 367]) and recollect all the relevant evidence again.

In this chapter, we explore the idea of having a system dynamically help users keep track of and organize information by leveraging the content they are browsing and the signals from their browsing behavior. Although we focus on the domain of programming due to strongly motivating prior work and ease of prototype development due to regularities of the programming context, our work may also generalize to other sensemaking contexts on the web. We instantiate this idea in a prototype system called Crystalline,[1] which is an extension to the Chrome web browser. Crystalline plays the role of a user's copilot and attempts to automatically identify and keep track of the options, criteria, and the corresponding evidence snippets from the web pages that a user has viewed, and organize the snippets into both list and tabular formats. To achieve this, Crystalline mines a variety of behavioral signals while a user browses the web, including scrolling patterns and mouse cursor actions, and employs natural language understanding techniques to automatically classify and organize the collected content. The goal is that users can focus more on reading and understanding web content while occasionally guiding the system when it makes mistakes. We conducted a user study to evaluate the usability and effectiveness of Crystalline compared to Unakite as a baseline, which found that developers are able to build comparison tables about 20% faster with a 60% reduction in operational cost without sacrificing the quality of the tables. In particular, it only requires around 12% of the total task completion time for participants to use the tool to build and maintain a table, compared to around 30% in the Unakite condition.

The primary contributions described in this chapter include:

- evidence that it is possible to automatically identify options, criteria, and relevant evidence from web pages that a user is browsing using a set of natural language understanding heuristics,
- a set of implicit behavioral signals that users exhibit when browsing the web which can be used for prioritizing and filtering the collected information,
- a prototype system called Crystalline that integrates the heuristics and signals to automatically collect and organize viewed information into list and comparison table views for subsequent decision making,

[1] Crystalline is named after rocks made up of interlocking crystals. It stands for **C**lipping **R**esulting in **Y**our **S**tructure as **T**ables **A**nd **L**ists **L**inked to **I**mplicit **N**otetaking **E**asily.

- an evaluation that offers empirical insights into the usability, usefulness, and effectiveness of those signals and the system.

4.2 Background and Design Goals

To ground our research, we build on the "Option-Criterion-Evidence" framework introduced in our Unakite system. We first briefly review the prior work on implicit behavioral patterns that people naturally exhibit while browsing the web that inspired our investigation. Then we discuss the design goals for the new Crystalline system.

4.2.1 Implicit Behavioral Signals When Using the Web

Prior research has investigated various implicit behavioral patterns that people exhibit when reading and interacting with content on a digital screen. One thread of research has explored using behaviors such as dwell time, cursor movements, clicks, scrolling patterns, and gaze positions as *implicit signals* to approximate user interest on web pages as well as search result relevance [85,139,140,158,169]. For example, Claypool et al. [85] had participants use a custom-built browser to surf the web and concluded that the time spent on a page, the amount of scrolling on a page, and the combination of time and scrolling had a strong correlation with explicit user interest. In addition, Hijikata [158] discovered that actions such as text tracing and link pointing are decent behavioral indicators for perceived interesting segments of web pages. Similarly, in the domain of web searches, Buscher et al. [61,62,63], Guo and Agichtein [139,140], and Huang et al. [169] demonstrated that eye tracking, as well as interactions like scrolling and cursor hovers, could accurately predict user interests in search results pages.

Building on such empirical understanding, we explore putting a combination of these implicit behavioral signals into use to approximate user visual attention in a working prototype. We used heuristics and pilot testing to devise mechanisms that translate the raw behavioral signals into numeric scores representing the "amount of attention" a user has given to a particular piece of online content. We then use these scores to filter out and rank the content of the evolving comparison table, further reducing the cost for developers to manually manage and prioritize collected information incrementally as they are searching and browsing.

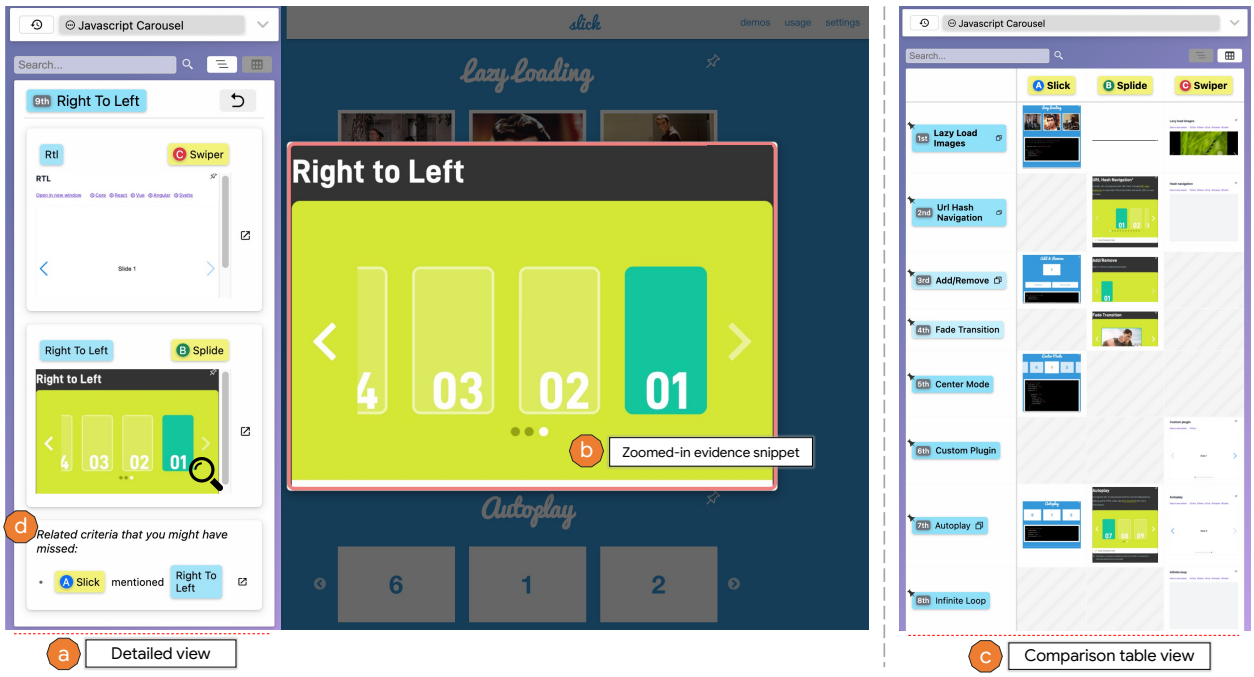


Figure 4.2: Additional Crystalline's user interfaces. Clicking on one of the criterion in the criteria pane (Figure 4.1-d) will enter a detailed view for that criterion (a), listing out all the collected evidence snippets organized by options. Users can zoom in on an evidence snippet (b) by moving the mouse cursor over it in the detailed view until the cursor becomes a magnifying glass. Crystalline will actively look for and remind users of evidence for the same or similar criteria from pages that users have visited but have not yet paid attention to (d). Finally, similar to Unakite [232], Crystalline offers a comparison table view (c) that summarizes the decision making space and the trade-offs among various options in detail.

4.2.2 Design Goals

In order to address the limitations of using Unakite as well as other similar sensemaking tools [37, 46, 75, 283] discussed in section 4.1, we formulated the following design goals:

- **Minimize the cost to collect information.** The system should attempt to automatically collect information in the background without the user's specific attention or direction. This will help users focus on the main task of reading and comprehending the content.
- **Actively filter, organize, and prioritize information.** The system should actively filter, organize, and prioritize the collected information that gets presented to the user and help the user avoid information overload.
- **Reduce the cost of incorrect automation support.** In cases where machine support is incorrect or undesirable, the system should allow users to easily recover from those mistakes [31, 164].

4.3 Crystalline

4.3.1 System Overview

Guided by prior work and our design goals, we designed and implemented Crystalline, a Chrome extension prototype to help developers automatically collect and organize information relevant to their decision making problems.

Users mainly interact with Crystalline through a **sidebar** (Figure 4.1-a) that is injected directly into every web page. As a developer opens and reads web pages, the sidebar will be updated with the automatically collected options (Figure 4.1-c) and criteria (Figure 4.1-d) in the *list view* (Figure 4.1-c & -d). The list view serves as a concise and glanceable outline that reflects one's exploration progress — what options one has encountered and what criteria one has looked into. Clicking on one of the criteria will enter a detailed view for that criterion (Figure 4.2-a), listing out all the collected evidence snippets organized by options; similarly, clicking on an option will enter the *detailed view* for that option, which lists all the related criteria and the corresponding evidence associated with that option. Details on how we currently implemented the automatic collection and organization features are discussed in section 4.3.2.

In addition, developers can also switch to the *comparison table view* (Figure 4.2-c) that summarizes the decision making space and the trade-offs among various options in detail. The order in which a criterion gets presented both in the list and the comparison table view are based on the *estimated importance* of the item to the user, which we approximate by the *amount of attention* a user has given to it. This, in turn, is derived from the user's implicit behavioral signals, which we will discuss in detail in section 4.3.2.2. To examine a particular piece of evidence in the detailed view or a comparison table cell, users can hover on it to zoom in (Figure 4.2-b), or click on it to *teleport* to the original web page and scroll position from where it was previously collected.

Similar to previous systems [163, 232, 293], the sidebar can be toggled in and out like a drawer by clicking the extension icon (Figure 4.1-h) or using a keyboard shortcut. Developers can passively monitor the sidebar as they are searching and browsing to make sure the system performs correctly, and quickly correct or dismiss the mistakes that the system makes. In addition, developers are free to hide the sidebar to have an unobstructed view of the web page, knowing that all the features for automatic information collection and organization are still running in the background, even if the sidebar is in the hidden state.

4.3.2 Detailed Design

We now discuss how the different features in Crystalline are designed and implemented, and how they support our design goals.

4.3.2.1 Collecting information about options and criteria

In Crystalline, we explore having the system *automatically* collect relevant information in the background without the user having to explicitly perform the action of collecting information.[2] This has the benefit of minimizing the distraction and cost of keeping track of information as an extra step in addition to thinking about the content on a web page, which, in turn, maximizes a user's attention to reading and understanding the content itself.

Specifically, Crystalline collects information about options, criteria, and their associated evidence snippets as discussed previously, which was reported by prior work as the key aspects developers look for when solving decision making problems [166, 207, 232]. Currently, to automatically recognize the *options*, Crystalline employs the following techniques: (1) it looks for the word or phrase between any instances of "vs." (or other variants like "v.s.", "versus", etc.) in web page titles and opening paragraphs and adds them as potential options. For example, the Medium.com article titled "Tensorflow vs Keras vs Pytorch: Which Framework is the Best?"[3] would yield "Tensorflow", "Keras", and "Pytorch" as three potential options; (2) it first runs noun phrase and entity extractions using the Google Cloud Natural Language API [135] on the web page title, section headers as well as the column and row headers of any HTML tables, then checks if the identified entities are mentioned in the titles of other visited pages. In addition, it also checks if the identified entities would frequently come up in each other's Google autocomplete results (the Google "vs" technique is described in [120, 233], which issues queries in the form of "`[option_name] vs`" to the Google Autocomplete API to get a list of autocomplete results that can be interpreted as potential alternatives to "`[option_name]`". Furthermore, Crystalline checks if the identified entities are mentioned repeatedly across the main content of the current web page. All potential options will go through a final deduplication process to produce the final list of options presented in the *options pane* (Figure 4.1-c) in the sidebar. We chose and tuned these heuristics based on our internal usage and pilot testing results. In the future, more

[2] At the time this work was executed, powerful large language models (LLMs), such as `GPT-4` and PaLM 2 that can perform such information extraction tasks with state-of-the-art performance weren't yet available. As a follow-up, the Selenite system from Chapter 7 investigated using `GPT-4` to perform such information extraction tasks more accurately and comprehensively.

[3] https://medium.com/@AtlasSystems/tensorflow-vs-keras-vs-pytorch-which-framework-is-the-best-f92f95e11502

advanced NLP techniques could be used to augment the current set of heuristics, for example, Chapter 7 introduces the idea of leveraging the powerful information extraction and reasoning capabilities of large language models [274] to extract options directly from the text content of a web page (see section 7.3.2.1 for more details).

Crystalline uses a similar set of heuristics to identify *criteria* from the web pages, with an emphasis on examining section headers and table headers (and entities extracted from them) rather than website titles. In this work and in the context of programming, we focus on using such heuristics to identify the criteria directly mentioned in the content, such as extracting "learning curve" from "React is widely considered to have quite a steep learning curve." We leave the extraction of implicit criteria for future work (such as leveraging the advanced reasoning capabilities of LLMs like `GPT-4` [274], as discussed in Chapter 7), which are more commonly seen in domains other than programming, such as extracting "price" from "I bought this mp3 player for almost nothing" [292].

Further, users can always edit the options and criteria names, delete unwanted options or criteria, or manually select and collect any text as either an option or a criterion using the popup menu (Figure 4.3) as a backup.

4.3.2.2 Organizing and prioritizing information

Not all options or criteria are equally useful to a particular developer. Prior work has suggested that a programming decision usually comes down to how well each option matches the developer's goals and criteria that he or she deemed important [127, 209, 211, 233, 281, 285, 295, 308]. In this chapter, we explore using the amount of attention that one pays to a particular criterion to approximate its perceived value or importance. To operationalize this, for each web page that a developer visits, Crystalline processes all the content blocks (HTML block-level elements, such as <p>, <li>, <pre>, and , etc.) to detect what options and criteria are associated with each block. Specifically, it prioritizes verbatim mentioning of options and criteria within a block, then possible options and criteria identified from section headers above the block, then web page titles. If no options are detected, the page title is used as a placeholder.[4]

Next, Crystalline tracks each triggering of five implicit behavioral signals (*copying content*, *text highlighting*, *clicking*, *cursor hovering*, and *content dwelling*) listed in Table 4.1 on any content block and translates it into a numeric score (using column 5). The final attention score A_c

[4]Similarly, informal testing conducted in Chapter 7 suggested that `GPT-4` is much more capable and reliable at detecting what options and criteria are associated with each content block.

Implicit Behavioral Signal	Selected References in Prior Research	Descriptions	Strength of indication of user attention	Score Function W
Copying content	Developers frequently copy sample code from the web to use in their own code [54, 149, 150]	Triggers when the user copies some text from a content block b. This typically happens when a developer copies sample code from web pages to try out in their own code.	Strongest	40 for each triggering
Text highlighting	People tend to highlight text while reading to help focus their attention [304]	Triggers each time when some text in a content block b gets selected. Triggerings where the selected text is shorter than 5 characters are disqualified.	Strong	20 for each triggering
Clicking	Clicking on content, such as widgets and links, is considered to be a decent behavioral indicator for perceived interesting elements on web pages [158]	Triggers when the user clicks on a content block b. This accounts for situations where the developer interacts with content on a page, such as live demo widgets. Clicks that are part of text highlighting are excluded.	Strong	20 for each triggering
Cursor hovering	People tend to use the cursor to guide their attention while reading web pages [79, 141, 158, 169, 299].	Triggers each time when the mouse cursor hovers over a content block b for at least 2 seconds. This accounts for situations where the developer naturally moves the mouse cursor onto the content that is currently being read to guide his or her attention [79, 168, 299]. However, a cursor hover triggering will be disqualified when the system detects an extended period of idling (2 minutes) without any user actions.	Weak	$0.5t$, where t is the duration (measured in seconds) of the cursor's stay within the bounds of content block b. The maximum score is 10. In our pilot testing, users rarely spend more than 10 seconds reading a text block.
Content dwelling	The longer some content stays visible, the more likely that the user is interested in it [85, 169].	Triggers each time when a content block b gets scrolled into and stays in the visible view port for at least 2 seconds. This indicates that the developer has at least paid attention to b. However, a dwell triggering during idling is disqualified.	Weak	$0.2t$, where t is the duration (measured in seconds) of content block b's stay in the visible browser viewport. The maximum score is 4. In our pilot testing, users rarely stay at one location for more than 10 seconds.

Table 4.1: Implicit behavioral signals used in Crystalline to track user attention. Column 1 lists the implicit signals; column 2 provides evidence from selected prior research on the efficacy of the signals; column 3 describes how the signals are used in Crystalline; column 4 indicates the relative strength of a signal in terms of predicting user attention; column 5 details the scoring function used to translate signal triggerings into numeric scores based on the relative signal strengths. The scoring functions were empirically determined through iterative pilot testing.

representing the amount of attention that a user pays to a particular criterion c is then calculated using equation (4.1):

$$A_c = \sum_{t \in T} I(t, c) \times W(t) \tag{4.1}$$

where T is the set of all implicit signal triggerings; t is a particular triggering; $I(t, c)$ returns 1 if t was triggered on a content block that is associated with the criterion c, and returns 0 otherwise; and $W(t)$ is the corresponding scoring function found in the last column in Table 4.1. The scoring functions were empirically determined through iterative pilot testing.

To accommodate various behavioral patterns exhibited by different users, we iteratively recruited four batches of participants with diverse backgrounds and job responsibilities both within our lab and externally. We followed a diary study approach [297] by monitoring their online

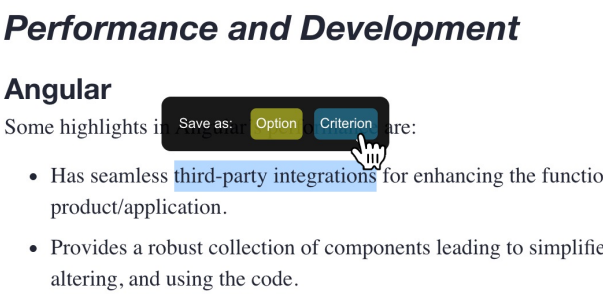


Figure 4.3: Using the selection popup menu to manually collect options and criteria.

searching and browsing behavior related to programming through a custom chrome extension that logs triggerings of the above behavior signals and ranks the importance of the associated content blocks accordingly (the initial score functions were determined through our heuristics). At the end of each sensemaking episode, we prompted them to review how well the system did in inferring what they thought was important, and tuned the score function heuristics accordingly (favoring recall over precision). We leave more advanced and adaptive scoring models for future work to investigate.

By default, the system shows the top 15 criteria ranked by decreasing attention scores in both the list and the table view. Users can use the "See More" and "See Less" buttons to adjust how many criteria that they would like to see at the same time (Figure 4.1-g). As the user browses more content and spreads his or her attention on different content blocks, the order of these criteria changes accordingly in real-time, which provides the user with an ambient awareness of what the system thinks are important. To provide users with the flexibility to override the system's ranking, they can right-click on a criterion and use the "pin this criterion" feature to pin it at the top (Figure 4.1-i). They can additionally specify their own order of preferences by dragging and dropping to reorder the criteria in the table view, which will automatically pin a criterion if it is not already pinned. Each time an implicit behavioral signal triggering is detected, Crystalline also collects the target content block as an evidence snippet, which is presented with its original styling [232] in the detail views and the comparison table view as mentioned above.

4.3.2.3 Managing connections and relationships.

One way for Crystalline to actively manage the relationships among the collected information is to automatically merge similar criteria together into *criteria groups* (indicated by a "multiple items" icon at the end, see Figure 4.1-e). To achieve this, we leverage transformer machine learning models such as Universal Sentence Encoder [70] and BERT [99] that can encode textual content into semantically meaningful vector representations called embeddings [128], i.e., two or

more semantically close pieces of content will also be close in the embedding vector space (measured by a distance metric, e.g., the cosine similarity distance between vectors [326]). Crystalline computes an embedding for every criterion as the average of its own embedding and its corresponding evidence snippet, and automatically merges criteria that are within a specified semantic distance threshold to each other into a group. For example, as shown in Figure 4.2-a, the system automatically merges "Right to Left" (taken from the option "Splide") and "RTL" (taken from the option "Swiper") together since they are semantically similar. The distance threshold was determined empirically through iterative pilot testing. This has the benefit of reducing clutter while helping users make connections among the information that they have seen, which is reported by prior work as one of the difficult steps during sensemaking and schematization [114, 288, 307]. In case the system fails to automatically group similar criteria together, users can use drag and drop to manually make the grouping. Similarly, users can easily split a criteria group by right-clicking on the group and hitting the "split this criteria group" menu item.

In situations where a user reads and investigates some criterion at one location, Crystalline will also actively look for evidence for the same or similar criteria from other pages that the user has visited (including the current page) but has not (yet) paid attention to according to the implicit signals. Crystalline will remind the user of the existence of this additional evidence through a red notification dot at the top right of a criterion (Figure 4.1-f) as well as in the detailed views (Figure 4.2-d). This then serves as an additional way for the system to help users uncover and manage unseen relationships among the information space, as well as a springboard for users to jump directly to the "overlooked" information for further investigation.

4.3.3 Implementation Notes

To produce the content embeddings, we used *bert-as-a-service* [99] and the `uncased_L-12_H-768_A-12` pre-trained BERT model to implement a REST API that the extension can query on-demand. The embedding calculations are known to incur significant computational costs and delays. Therefore, to ensure a smooth user experience, they are better suited to run on a remote server with the necessary resources rather than locally in an end-user's browser.

Unlike other systems [107, 286] that help users find more information from new sources, Crystalline only collects information from the web pages that a user has explicitly visited. This is an intentional design choice we make in the current implementation: the major role of Crystalline is to remove the burden for users to actively keep track of relevant information that they have personally seen and investigated so that it is easier for them to revisit and recall.

4.4 Evaluation

We conducted an initial lab study to evaluate the usability of the Crystalline system in helping developers collect and organize information.

4.4.1 Participants

We recruited 12 participants (7 male, 5 female) aged 22-35 (μ = 27.6, σ = 3.7) years old through emails and social media. The participants were required to be 18 or older, fluent in English, and experienced in programming. Participants had on average 6.9 years of programming experience, with half of them currently working or having worked as a professional developer and the rest having programming experience in universities.

4.4.2 Procedure

The study was a within-subjects design, where participants were presented with two tasks and were asked to complete one of them using Unakite (baseline condition) and the other using Crystalline (experimental condition), in a counterbalanced order. For each task, participants were presented a programming decision-making problem, a set of four web pages, some necessary background of the problem, and a list of three options available to solve the problem that they were required to investigate. The provided web pages were either documentation pages of specific options or comprehensive review articles reviewing several options together. Participants were instructed to read through the provided web pages, and use either Unakite or Crystalline to collect and organize information into a comparison table containing all the given options and at least 8 different criteria in the order of their perceived importance. We imposed a 20-minute limit per task to keep participants from getting caught up in one of the tasks. However, they were instructed to inform the researcher when they have collected 8 criteria as well as the associated evidence. If they wished to continue beyond this checkpoint, they were allowed to, until they felt like they could make no further progress. Specifically, the two tasks were to use the corresponding system in each condition to build a comparison table of:

- *(A)* Choosing a JavaScript carousel library to build a photo-sharing web application. The available options were: Splide.js [22], Slick [21], and Swiper [23].
- *(B)* Choosing a front-end framework to implement a basic personal portfolio website. The available options were: React.js [109], Angular [134], and Vue.js [24].

We chose Unakite over other commercially available tools such as Google Docs as the baseline condition because: 1) it can be easily used to capture richer contexts such as formatted text (example code), images, and links; 2) similar to Crystalline, it also provides a sidebar that allows participants to view and organize the collected information directly rather than switching context over to another browser tab or application to paste in and structure information; and 3) Unakite was shown to be easy to learn and use in prior research and incurs significantly less overhead cost than using Google Docs [232] (as discussed in Chapter 3, section 3.4.1.2).

In addition, rather than letting participants search for their own pages to research, we provided them with the predefined set of pages to ensure a fair comparison of the results, and since helping to find relevant web pages is not a goal of Crystalline. Requiring participants to only read the predefined pages (each contains on average 7 screenfuls of content) also helps ensure that the two tasks are of roughly equal difficulty in terms of reading and cognitive processing effort. Furthermore, to ensure realism and participant engagement, the tasks were selected based on actual questions asked and discussed on programming forums and websites. We specifically simplified the requirements and background of task B to match that of task A, since otherwise, choosing a JavaScript framework (e.g., to build interactive industry-level web applications) would arguably be more substantial and involve deeper and much more careful comparisons and team discussions that are beyond the scope of this lab study. In fact, as shown in section 4.5.1 there was no significant difference by task.

Each study session started by obtaining consent and having participants fill out a demographic survey. Participants were then given a 10-minute tutorial showcasing the various features of Unakite and Crystalline and a 10-minute practice session on both systems before starting. At the end of the study, the researcher conducted a survey and an interview eliciting subjective feedback on the Unakite and Crystalline experience. Each study session took approximately 60 minutes, using a designated MacBook Pro computer with Chrome, Unakite and Crystalline installed. All participants were compensated $15 for their time.

4.5 Results

4.5.1 Quantitative Results

All participants were able to complete all of the tasks in both conditions, and nobody went over the pre-imposed time limit. Figure 4.1, together with Figure 4.2, shows an example table built by one of the participants in the study for task A.

	Manually select information and capture	Rename an option / criteria	Delete an option / criteria	Manually put information snippets into the table	Remove a snippet from the table	Merge criteria into groups	Split criteria groups	Pin or reorder criteria	**Overall**
Task A	27.0 (6.42)	1.67 (1.97)	0.67 (1.03)	16.5 (5.43)	0.50 (0.84)	N/A	N/A	6.00 (2.19)	52.3 (13.7)
Task B	26.2 (5.56)	1.83 (1.60)	1.50 (1.38)	14.5 (5.28)	0.33 (0.82)	N/A	N/A	6.00 (1.79)	50.3 (14.3)
Average	26.6 (5.74)	1.75 (1.71)	1.08 (1.24)	15.5 (5.21)	0.42 (0.79)	N/A	N/A	6.00 (1.91)	51.3 (13.4)

(a) Unakite condition

	Manually select information and capture	Rename an option / criteria	Delete an option / criteria	Manually put information snippets into the table	Remove a snippet from the table	Merge criteria into groups	Split criteria groups	Pin or reorder criteria	**Overall**
Task A	0.83 (0.75)	2.17 (1.17)	0.50 (0.84)	0.17 (0.41)	0.33 (0.52)	2.33 (0.82)	0.83 (0.75)	5.33 (1.97)	12.5 (3.02)
Task B	1.00 (1.26)	1.67 (0.82)	0.50 (0.55)	0.33 (0.52)	0.33 (0.52)	1.83 (0.75)	0.67 (0.82)	5.50 (2.74)	11.8 (3.31)
Average	0.92 (1.00)	1.92 (1.00)	0.50 (0.67)	0.25 (0.45)	0.33 (0.49)	2.08 (0.79)	0.75 (0.75)	5.42 (2.27)	12.2 (3.04)

(b) Crystalline condition

Table 4.2: Statistics for the average number of interactions performed by users to perform the tasks in the user study. Standard deviations are included in the parentheses.

To examine how Crystalline performs compared to the baseline Unakite condition, we measured the time it took for participants to finish each task. A two-way repeated measures ANOVA was conducted to examine the within-subject effects of condition (Crystalline vs. Unakite) and task (A vs. B) on task completion time. There was a statistically significant effect of condition ($F(1,20) = 8.06$, $p = 0.01$) such that participants completed tasks significantly faster (21.6% faster) with Crystalline (Mean = 611.8 seconds, SD = 144.6 seconds) than in the Unakite condition (Mean = 780.3 seconds, SD = 137.6 seconds). There was no significant effect of task ($F(1,20) = 0.11$, $p = 0.74$), indicating the two tasks were indeed of roughly equal difficulty. These results suggest Crystalline helped participants build up comparison tables faster overall, even the majority of their time was necessarily spent reading through the material in both conditions.

To account for this reading time, we also compared the *overhead cost* (see section 3.4.1.2) of using both tools to collect and organize information. For the Crystalline condition, we calculated the overhead cost as the portion of the time participants spent on directly interacting with Crystalline (scrolling through the list and table view to examine the evidence collected so far, splitting and merging criteria, pinning important criteria, manually collecting information, etc.) out of the total time they used for a task (vs. reading and comprehending the web pages). Similarly, in the Unakite condition, the overhead cost was calculated as the percent of time participants spent on directly using Unakite features (selecting and collecting snippets, drag and dropping them into the comparison table, etc.), in the same way as was done to compare Unakite to Google Docs.

A two-way repeated measures ANOVA was conducted to examine the within-subject effects of condition (Crystalline vs. Unakite) and task (A vs. B) on overhead cost. There was a statis-

tically significant effect of condition ($F(1,20) = 77.5$, $p < 0.001$) such that the overhead cost was significantly lower (almost 60% lower) in the Crystalline condition (Mean = 11.6%, SD = 0.04) than in the Unakite condition (Mean = 28.4%, SD = 0.07). Again, there was no significant effect of task ($F(1,20) = 0.53$, $p = 0.48$)). Thus, using Crystalline resulted in reduced overhead costs of collecting and organizing information.

To gain deeper insights into *why* the overhead cost was significantly lower in the Crystalline condition, we tallied the number of interactions performed in each task while collecting and organizing information to build the comparison tables (Table 4.2). Here, we notice that the majority of interactions in the Unakite condition are to manually collect information snippets (on average 26.6 times) and place them into the comparison table (on average 15.5 times). In contrast, in the Crystalline condition, the majority of interactions are to merge criteria into groups (on average 2.08 times) and pin or reorder the criteria in the table (on average 5.42 times). This suggests that, to some extent, Crystalline has transformed the previously active capturing and organizing work into passive monitoring and error-fixing, which explains the lower overhead cost.

In the survey, participants reported (in 7-point Likert scales) that they thought the interactions with Crystalline were understandable and clear (Mean = 6.17, SD = 0.39), Crystalline was easy to learn (Mean = 6.08, SD = 0.79), and they enjoyed Crystalline's features (Mean = 6.25, SD = 0.45). In addition, compared to Unakite (Mean = 5.75, SD = 0.45), they thought using Crystalline (Mean = 6.08, SD = 0.29) would help them solve programming problems more efficiently and effectively, and would recommend Crystalline (Mean = 6.17, SD = 0.58) over Unakite (Mean = 5.58, SD = 0.51) to friends and colleagues doing programming work, both differences were statistically significant under paired t-tests.

4.5.2 Qualitative Observations

4.5.2.1 Usability and usage patterns

Overall, participants appreciated the increased efficiency afforded by various Crystalline features. Many (9/12) mentioned that the perceived workload to collect and organize what they have investigated was minimal, saying that "*I feel like I got a table for free*" (P3), "*the fact that I can see what I've paid a lot of attention to automatically bubbles up to the top is quite magical*" (P9), and "*It feels as if I was sitting in the passenger seat and not having to do all the steering and maneuvering*" (P7). Some (3/12) participants also reported having taken advantage of the overlooked information reminder feature (Figure 4.2-d) to guide their research. Furthermore, participants reflected that Crystalline relieves them of the burden of trying to anticipate the value of a partic-

ular piece of information before collecting it since "*the important bits will eventually be at or near the top, hopefully*" (P12), and they could "*focus on reading the page itself and not context switch to bookkeeping mode again and again*" (P5).

However, some did voice concerns about the system's ability at the beginning of the tasks, arguing that they were "*skeptical if it will actually collect the right things*" (P1), and reported that they would "*skim through the list view and the table view quite frequently at the beginning*" (P7). However, as they progressed through the tasks, their confidence in Crystalline increased, and they only occasionally checked the sidebar. We observed that three of the 12 participants ended up not examining and editing the system's output until they felt like they had finished reading and processing all the given pages, and they made minimal edits to the results.

4.5.2.2 Working with machine suggestions

Participants generally thought that the benefits of automating the collection and organization process outweighed the costs of dealing with occasional unhelpful machine suggestions, such as incorrectly merging criteria together or prioritizing unimportant criteria at the top of the list. For example, P7 reflected, "*it feels like a mind reader. I know it's not perfect, but I also don't expect it to be, and would actually prefer occasionally peeking into what it's been doing and fixing whatever that's not correct than grabbing everything by myself all the time.*"

Some did raise concerns about the ordering of criteria getting changed too frequently ("*they [the criteria] were jumping around*", P7) at the beginning. This is likely due to the fact that users were skimming through a web page without paying particular attention to anything at the beginning, causing their attention scores to be relatively indistinguishable. For future iterations of the system, we could experiment with less frequent UI update intervals under these circumstances so it would cause less distraction.

4.5.3 Evaluation Discussion

Similar to what was reported in prior work [293], since our participants were not explicitly told how the system worked to automatically collect and rank information, they had to form their own mental models and hypotheses about how the system works and how they could affect it with their behavior. For example, P8 noticed that "*it looks like if I spend a little bit more time on a particular place on a page, the corresponding criterion would get picked up and bumped up quickly; and if I click on that part a bunch of times, which happens to be what I typically would do when I try to focus my attention on something now that I'm thinking about it, it's [the corresponding criterion] going to go up even faster.*" This suggests that our implicit signals were working, and further, that

with experience users might adapt to *explicitly* steer the system towards their goal of collecting and prioritizing information, resulting in, to some extent, a mixed-initiative collection approach that still would require much less effort than the baseline methods. Future research could explore the costs and benefits of a wide variety of interactions and signals that lie on the spectrum between implicit behavioral signals to full manual direct manipulations, and any differences caused by directly instructing users about the implicit signals being used.

Though the current version of Crystalline mainly focuses on reducing the cost for developers to collect and organize information, which was exactly what we tested in the lab study, we were also interested in making sure that the *quality* of the comparison tables built using Crystalline does not degrade as seen in other automation scenarios [138, 324]. Since there is not a gold standard comparison table, we evaluated the correctness of Crystalline's automatic approaches by how much editing participants had to do in order to fix Crystalline's mistakes and make sure that all the content in the table was eventually filled out and ranked correctly according to their understanding as per the study protocol. As shown in Table 4.2(b), participants only had to perform on average 12.2 edits to the automatically generated comparison tables, compared to the 51.3 actions that they had to manually perform in the baseline Unakite condition (the difference is statistically significant, $p < 0.01$). Among these, edits that are related to collecting information, such as manually selecting information and capture (0.92 times), renaming (1.92 times), and deleting information (0.50 times) were minimal, suggesting that our combination of NLP and behavioral signal heuristics was working effectively to collect information that the users thought was important. However, participants pinned or reordered the criteria that were automatically ranked by Crystalline on average 5.42 times (SD = 2.27 times). One possible explanation is that the universal scoring functions (in Table 4.1) did not necessarily apply to every single participant, suggesting the need for a more sophisticated and personalized scoring mechanism in future iterations of Crystalline and systems that leverage signals from users' natural browsing behavior.

In addition, we asked and coded their opinions about *using* these tables as if they were the subsequent developers trying to *understand* the design rationale. In general, participants were excited about using comparison tables automatically built by Crystalline. For example, P10 highlighted scenarios where Crystalline would be useful for his own purposes, saying that "*it's sort of like a never-erased whiteboard that would most likely help me remember what I looked at three months ago.*" In addition, some reflected that compared to having no clue of why a decision was made in a particular way in the first place, they would appreciate at least having access to a Crystalline table even if it was not actively monitored and maintained during the initial developer's sensemaking process. For example, P4 said: "*I think being able to read something like this [Crys-*

talline table] is going to make a big difference when you're banging your head against the wall trying to understand why this particularly old API was chosen, I mean, especially when the guy who wrote the code was long gone, I could at least 'read a transcription of his mind' in some sense." Here, we see preliminary evidence that our approach of automatically collecting and organizing information on behalf of developers is useful and valuable. We leave the formal evaluation of the quality of fully automatically built comparison tables with possibly more advanced versions of Crystalline for future work.

4.6 Limitations

Currently, Crystalline works best on a limited set of web pages in the programming domain, including documentation pages that are dedicated to a particular library or a set of APIs, as well as review articles or question answering pages that discuss and compare several options together. We chose to optimize for these types of web pages in the current prototype as they are reported in prior work [166, 232] as well as our formative discussions with developers as some of the most frequently consulted programming resources when it comes to making decisions. However, the performance reported on the web pages used in the study is not necessarily representative of how Crystalline would operate even on web pages of these types for users in general. In addition, Crystalline currently relies heavily on the overall structure of the web pages being standard, meaning that a page uses HTML tags appropriately according to their semantics (e.g., enclosing headers and list items in `<h>` and `<li>` tags rather than wrapping everything with `<div>` tags) and that there is a strong semantic coherence between a section header and its corresponding content. Though this is sufficient to demonstrate the idea of automatic collecting and organization and the benefits they offer, future research is needed to make Crystalline-style tools work on a more diverse set of web pages, as well as how to be clear upfront about its limitations in parsing web pages that do not follow appropriate web standards.

In addition, our lab study has several limitations. Given the short amount of training and practice time participants had, some might not have been able to fully grasp the various features of Crystalline, or they might have been confused about what Unakite (the baseline system) has to offer. The study tasks might not be what participants typically encounter in their daily work, depending on whether they are in a position to make decisions, and thus they may not be equipped with the necessary motivation or context that they would otherwise have in real life. We mitigate these risks in the study setup by: 1) having participants perform a practice task for each condition simulating what they would have to do in the real tasks; 2) choosing the study tasks

based on actual questions that are discussed by developers on Stack Overflow and other popular programming community forums; and 3) providing participants with sufficient background information and context to help them get prepared. In fact, 7 out of 12 participants reported that the tasks were indeed similar to what they would deal with in their daily work. One further address these limitations in the future by having developers use Crystalline on their own work and personal projects, which would provide them with sufficient motivation as well as experience with Crystalline enriched over time.

Furthermore, the overhead cost measurement in the study could be conservative, as we did not account for the time participants spent simply glancing or looking at the sidebars without any explicit interactions with it. However, from our observations during the study, participants rarely spent any extended time doing this. Nevertheless, we would like to take advantage of more advanced tools such as eye tracking [49, 278, 279, 299] in the future to more accurately account for the proportion of time when a participant's gaze is fixated on the user interface of the tools rather than on actual web content.

4.7 Discussion and Future Work

Through designing and evaluating Crystalline, we gained deeper insights into the benefits and trade-offs of automatically collecting and organizing information for developers as they make sense of the web to make programming decisions. This motivates some ideas for future work.

While Crystalline's approach provides developers with an inexpensive way of capturing knowledge in the browser, it represents only one piece of a larger puzzle of how to support a developer's everyday work that involves sensemaking and decision making. One dimension to characterize this is that developers also frequently perform activities outside their browsers, such as in IDEs, code editors [310], command-line interfaces [81], literate programming notebooks [186, 188], or threads of discussions during formal or informal meetings [378]. Further research would be needed to understand how to collect and organize information from these sources as well as how to integrate them together to provide a more comprehensive picture of the decision making context. Another dimension that is relevant is the lifecycle of the knowledge captured via systems like Unakite and Crystalline. Early evidence from the user study has suggested there is a benefit of Crystalline's organization from the perspective of a subsequent developer who may need to understand a previous developer's decision. Future research could investigate how well developers are able to understand and potentially reuse these automatically assembled knowledge artifacts, possibly without any manual interventions from the initial knowledge authors,

which could, in turn, eliminate the starting cost associated with initial knowledge creation [114] and unlock the virtuous cycle of accelerated programming knowledge reuse [114, 233].

Though the current set of mechanisms for deriving the importance of criteria from implicit behavioral signals generally works well for the setting of this research, there could be situations where a user's default browsing behaviors and patterns fall outside the limited set of signals and heuristics that Crystalline is currently looking for. For example, a user might not have the habit of unconsciously using the cursor as a reading guide or might not interact with the page at all while reading, which would render the tracking of some of the behavioral signals moot. In addition, users could exhibit different or additional behavior patterns when generalized to other tasks domains that involve information-backed decision making, such as comparison shopping, trip-planning, etc. [73, 145]. For example, when interacting with a map view to find the best local dining option, a user may frequently pan around and zoom (in and out) to view different restaurants, and both the duration of stay on a particular restaurant and how many times it is viewed back and forth could be leveraged to approximate the user's interest and investment of effort. One way to address these concerns is to leverage a more diverse set of behavioral signals and potentially signal combinations, such as scrolling, mouse panning, zooming, eye tracking [110, 111, 278, 279], and facial gestures tracking [189, 333] to collect a more accurate picture of what users are seeing on screen. Another future direction that could be fruitful is to take a machine learning approach instead of the current rule-based approach for approximating content importance using behavioral signals. Specifically, we could leverage recent advances in crowdsourcing and labeling [76, 83, 96, 328] to log, annotate, and construct a large-scale data set that maps a variety of behavioral signals to the perceived importance of content blocks that they are triggered on, and train on this data set to obtain scoring functions that would work more widely. Alternatively, an online learning approach could also be promising, where the system continuously learns, adapts, and improves from an individual user's behavior over time, as suggested by Horvitz [164].

Chapter 5

Wigglite: Lightweight Gestures for Collection and Triage

Consumers conducting comparison shopping, researchers making sense of competitive space, and developers looking for code snippets online all face the challenge of capturing the information they find for later use without interrupting their current flow. In addition, during many learning and exploration tasks, people need to externalize their mental context, such as estimating how urgent a topic is to follow up on, or rating a piece of evidence as a "pro" or "con," which helps scaffold subsequent deeper exploration. However, current approaches incur a high cost, often requiring users to select, copy, context switch, paste, and annotate information in a separate document without offering specific affordances that capture their mental context.

In this chapter, we introduce a new interaction technique called "wiggling," which can be used to fluidly collect, organize, and rate information during early sensemaking stages with a single gesture. Wiggling involves rapid back-and-forth movements of a pointer or up-and-down scrolling on a smartphone, which can indicate the information to be collected and its valence, using a single, lightweight gesture that does not interfere with other interactions that are already available. Through implementation and user evaluation, we found that wiggling helped participants accurately collect information and encode their mental context with a 58% reduction in operational cost while being 24% faster compared to a common baseline.

This chapter is modified from the following published paper: Michael Xieyang Liu, Andrew Kuznetsov, Yongsung Kim, Joseph Chee Chang, Aniket Kittur, and Brad A. Myers. 2022. "Wigglite: Low-cost Information Collection and Triage." *In Proceedings of the 35nd Annual ACM Symposium on User Interface Software and Technology (UIST '22). Association for Computing Machinery, New York, NY, USA, 67–80.*

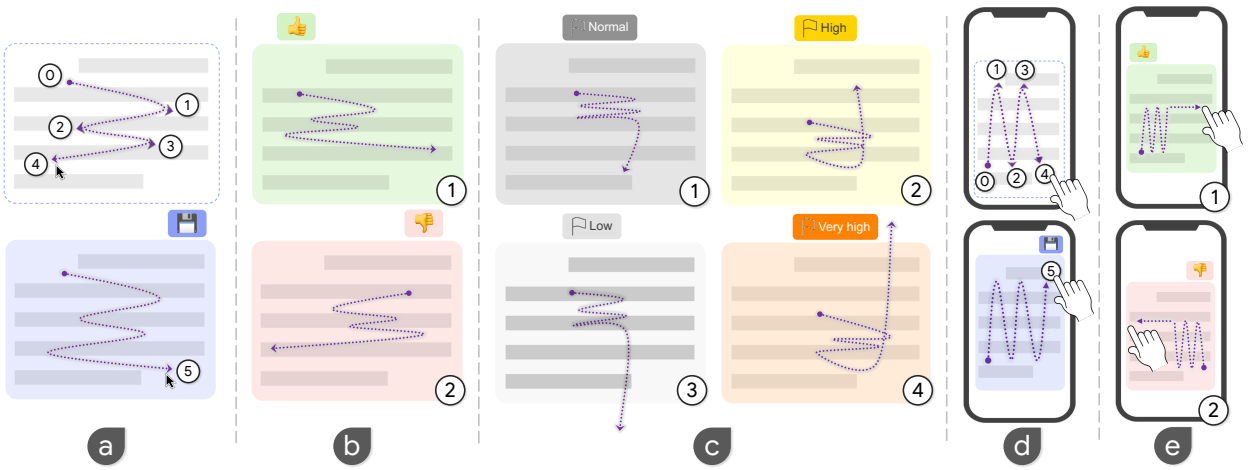


Figure 5.1: We introduce the "wiggling" technique: rapid back-and-forth movements of a mouse pointer on desktop (a) or a finger on mobile devices (d) that do not require any clicking to perform, yet are sufficiently accurate to select the desired content, while at the same time supporting an optional and natural encoding of valence rating (positive to negative) (on desktop: b1-2, on mobile: e1-2) or classification of priority (to facilitate triage) (c1-4) by ending the wiggle with a swipe in different directions.

5.1 Introduction

From consumers researching products, to patients making sense of medical diagnoses, to developers looking for solutions to programming problems, people spend a significant amount of time on the internet discovering and researching different options, prioritizing which to explore next, and learning about the different trade-offs that make them more or less suitable for their personal goals [73,74,75,193,232,233]. For example, a YouTuber seeking to upgrade her vlogging setup may learn about many different camera options from various online sites. As she discovers them, she implicitly prioritizes which are the most likely candidates she wants to investigate first, looking for video samples and technical reviews online that speak either positively or negatively about those cameras. Similarly, a patient might keep track of different treatment options and reports on positive or negative outcomes; or a developer might go through multiple Stack Overflow and blog posts to collect possible solutions and code snippets relevant to their programming problem, noting trade-offs about each along the way.

While the number of options, their likely importance, and evidence about their suitability can quickly exceed the limits of working memory, the high friction of externalizing this mental context means that people often still keep all this information in their heads [72, 159, 232, 247, 337]. Despite the multiple tools and methods that people use to capture information, such as copying and pasting relevant texts and links into a notes app or email [52], taking screenshots and photos [335], or using a web clipper [107], collecting web content and encoding a user's mental context about it remains a cognitively and physically demanding process involving many different

components: just the collection component itself involves deciding what and how much to collect, specifying the boundaries of the selection, copying it, switching context to the target application tab or window, transferring the information into the application where it will be stored [113], causing frequent interruptions to the users' main flow of reading and understanding the actual web content [159,193,247], especially on mobile devices [72,145]. In addition, components such as prioritizing options by importance result in additional overhead to move or mark their expected utility, which can change as users discover new options or old assumptions become obsolete. When further investigating each option, to keep track of evidence about its suitability, a user further needs to copy and paste each piece of evidence (e.g., text or images from a review or link to a video) and annotate it with how positive or negative it is relative to the user's goals.

Beyond the cognitive and physical overhead of collecting content and encoding context, prior work suggests that for learning and exploration tasks, people are often uncertain about which information will eventually turn out to be relevant and useful, especially at the early stages when there are many unknown unknowns [41, 72, 114, 145]. This could further render people hesitant to exert effort to externalize their mental context if that effort might be later thrown away [124, 232]. One relevant example is Kittur's Clipper [192, 193], which proactively prompted people to specify the "valence" (rating of good or bad) of an option (e.g., a specific camera) measured on a particular dimension (e.g., autofocus capability) as they collected information. Even though this elicitation of the mental model was done in situ and after much optimization of the interaction, it still required significant cognitive and physical effort and interruption, which prevented its widespread adoption. Other web clipping tools offer even less scaffolding for encoding mental context, typically only supporting a catch-all notes field that people rarely know how to take advantage of [88].

To summarize, we frame a fundamental sensemaking challenge for people trying to research and make decisions online as the high friction involved in capturing: (1) the content that they want to keep track of, which can range from a word, a phrase, an image, to a paragraph or multiple blocks of mixed multimedia content, (2) which option or topic that content corresponds to and its perceived priority for further investigation (which is called "triaging"), and (3) whether the evidence they find about that option or topic is positive or negative regarding its suitability for the user's goals (which is called "valence") [232].

Our vision in this work is to create a technique that reduces the friction for the transfer of a user's internal mental judgments while they are processing information into an external system that will capture those judgments and scaffold sensemaking and exploration. While it is a challenge for the cost of this transfer to be zero, we aim to reduce the overhead significantly

by exploring a new class of gestures for this purpose based on "wiggling:" rapid back-and-forth movements of a pointer that do not require any clicking to perform, yet are sufficiently precise to accurately select the desired content, while at the same time supporting the optional and natural encoding of valence rating (positive to negative) and classification of priority (to facilitate triage) to the collected information (see Figure 5.1). In addition, this technique does not conflict with typical existing interactions (like selecting text or clicking on hyperlinks) and can be extended to other device form factors such as touchscreens and mobile. The rating and classification can be applied by ending the "wiggle" with a swipe in different directions (see Figure 5.1-b,-c,-e).

We instantiate this class of wiggle-based gesture in an event-driven JavaScript library and a prototype system called Wigglite[1], which builds on top of an existing information and task management application called Skeema that already supports *clipping* and assigning *valence* to general web content as well as organizing them into *topics* with *priorities*. Wigglite consists of a Chrome extension and a mobile application, which enables users to capture and classify information fluidly while searching and browsing. To combat the issue of potentially collecting too much information, the system enables users to easily filter and sort the collected information based on the encoding that the users applied at collection time (or later).

In a performance evaluation on using wiggling for content selection, we found that wiggling is overall 35% faster on desktop and 40% faster on mobile, more accurate, and requires less physical and mental load to perform when compared to conventional selection when selecting blocks of content, especially on mobile devices. In addition, in a lab evaluation with participants, we found that using Wigglite to collect and triage information incurs 58% less overhead cost to perform without sacrificing operational accuracy. In addition, participants generally preferred the wiggling techniques over Skeema alone due to its easiness and naturalness to perform as well as its ability to encode their mental contexts in an organic way.

The primary contributions described in this chapter include:

- A novel class of wiggle-based gestures that are cognitively and physically lightweight to perform to collect information, and can simultaneously encode aspects of users' mental context,
- A prototype event-driven JavaScript library that implements such gestures and runs in web browsers,

[1]Wigglite stands for **W**iggling for **I**nformation **G**athering and **G**enerating **L**ightweight **I**mpressions for **T**riage and **E**ncoding.

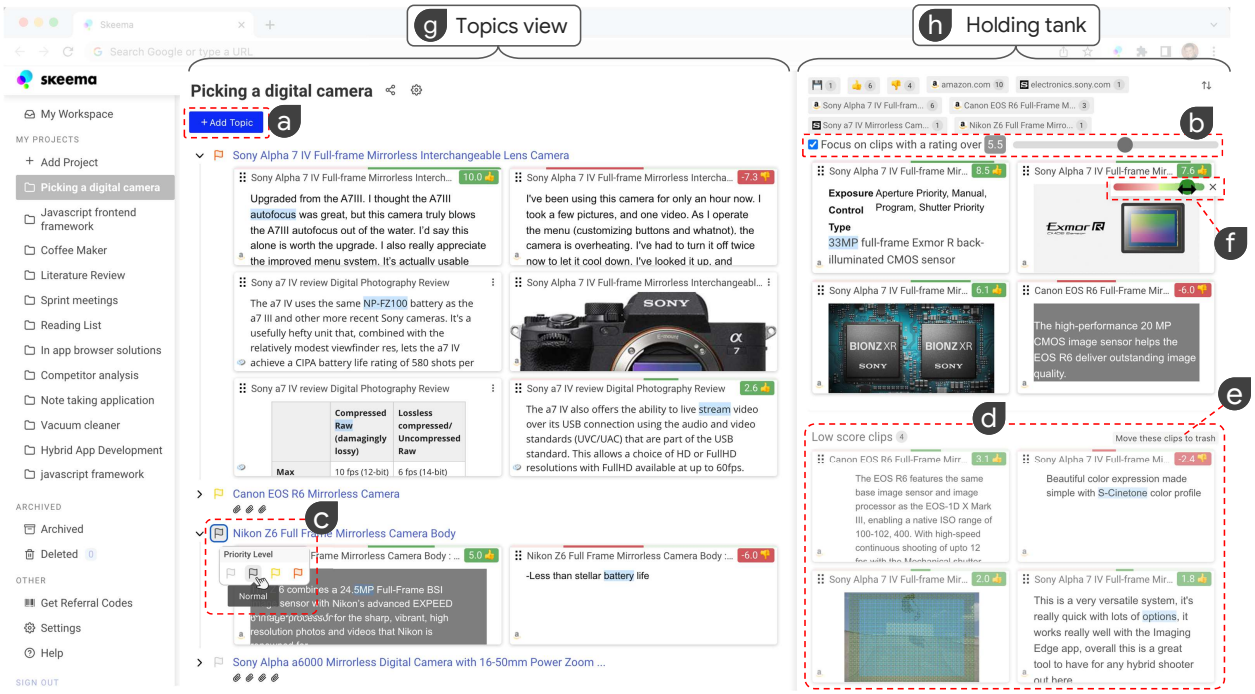


Figure 5.2: Wigglite's UI built on top of SKEEMA. On the left is the topics view (g) where users can create a topic (a) as well as change its perceived priority (c). On the right is the holding tank (h) that holds the collected information, in which users can filter out information with a lower rating using the slider (b). As a result, clips with rating scores lower than the set threshold would be automatically grouped together at the end and grayed out (d), and users can easily archive or put them in trash by clicking a button (e). In addition, users can manually adjust the valence rating of an information clip (f).

- Wigglite, a prototype system that takes advantage of the wiggle-based gestures to enable information capturing and classification during sensemaking that works on both desktop and mobile devices,
- A performance evaluation showing that compared to conventional methods, using wiggling to select content is faster and more accurate, both on desktop and mobile devices,
- A lab evaluation that offers empirical insights into the usability, usefulness, and effectiveness of the Wigglite system.

5.2 Related Work

5.2.1 Recognizing and Using Gestures

The wiggle gesture we use in Wigglite (as shown in Figure 5.1) has a similar form to a scratch-out gesture in some previous systems used for undo [365], edit [305], or delete. Wiggling has also been used by some window managers, for example, Microsoft Windows 7 in 2009 introduced

"Aero Shake" [342] where grabbing the title bar with the mouse and shaking the window left and right minimizes all other windows, or restores them. However, these gestures all require that the mouse button first be depressed, while our approach, on the contrary, is specifically designed to work when with none of the mouse buttons are depressed. In addition, macOS has an accessibility feature that supports shaking the mouse to make the size of the pointer much larger to help locate the pointer [334]. Importantly, our testing shows that those features do not interfere with our browser-based implementation of wiggle-based gestures.

Over the years, many complex gesture *recognizers* have been developed, such as the Rubine recognizer [305], which extracts multiple features from a trajectory and uses a linear classifier for recognition. However, these parametric recognizers are difficult to control with respect to the variances in gestures to be supported. Another approach is template-based gesture recognition, such as the $1 recognizer [366] and the Protractor recognizer [226], which compare new trajectories to the pre-defined gesture templates, and are more lightweight without sacrificing too much accuracy. However, these recognizers can be both time and resource intensive, especially on mobile devices where the computing power and resources are usually limited. In our work, we built a heuristics-based ad-hoc recognizer (see section 5.4), allowing the system to perform real-time eager recognition [306] without impacting the performance of other UI activities on both desktop and mobile devices. In addition, building on prior evidence that people can accurately perform swipes to as many as eight different directions [65, 203], we support ending the wiggle gesture with a directional swipe to further classify the collected information or encode people's mental context in situ.

5.3 Background and Design Goals

To ground our research, we build on an existing information and task management system called Skeema. First, we briefly describe Skeema and its features related to the context of this work, then discuss the design goals and processes for the wiggling gesture for the new Wigglite system.

5.3.1 The Skeema system

Skeema is a Chrome browser extension designed to support people's need to collect and organize information and manage their tabs during online sensemaking. Different from general web clippers that typically only support saving entire pages of web content into an individual note within a notebook [107], Skeema enables people to save an arbitrary amount of web content as

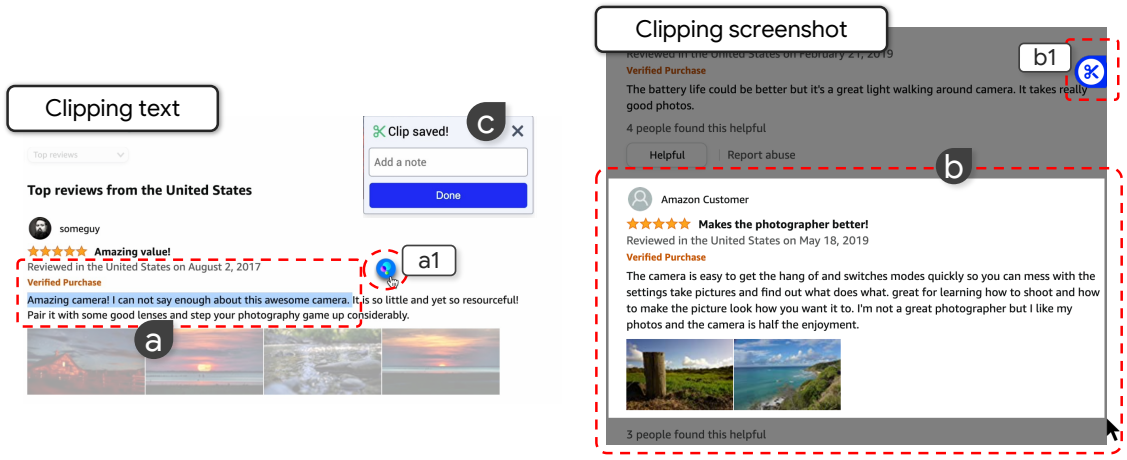


Figure 5.3: Two clipping mechanisms that SKEEMA supports: clipping text (left) and clipping screenshot (right).

information clips (Figure 5.5-c) into a holding tank (Figure 5.2-h), and later organize them into topics in the topics view (Figure 5.2-g). For clipping, SKEEMA offers two methods:

- **Clipping text**: Users can select arbitrary content using the cursor and click the clipping button that pops up to collect the selected texts (see the left part of Figure 5.3).
- **Clipping screenshot**: Users can use the screenshot feature to drag out a bounding box to save the desired content (see the right part of Figure 5.3).

To help users express whether a piece of evidence that they collected is positive or negative with regard to their own goal, SKEEMA allows users to add a valence rating from -10 to +10, with negative values indicating a "con" and denoted by a "thumbs-down" emoji and positive values indicating a "pro" and denoted by a "thumbs-up" emoji (see Figure 5.5-c1).

SKEEMA allows users to organize information into thematically related topics in the topics view (Figure 5.2-g). To achieve that, users need to manually create a topic (Figure 5.2-a), enter a name, and drag the desired information cards from the holding tank and drop it into the topic. Users can also set priority to a topic to indicate its perceived utility and how much they want to follow up on it, which defaults to be "Normal", but can also be set to "Low", "High", or "Very high" (Figure 5.2-c).

Although SKEEMA has the support for collecting finer-grain content (which research has shown to be the unit of information that people usually think in and work with during sensemaking [247, 314]), there is still a high cost in specifying the collection boundary and adding ratings and priorities to the collected information and topics (which users would have to switch to the SKEEMA tab to do). In addition, clipping text in SKEEMA loses the text's original CSS styling, which might be helpful for quicker recognition later on [232], and SKEEMA does not gracefully support

collecting consecutive blocks of mixed content (e.g., consumer review text of a camera followed up some sample photos, such as shown in Figure 5.3-b).

5.3.2 Design Goals for Low-cost Information Capturing and Triaging

Guided by prior work and well as the limitations of Skeema discussed above, we set out to provide an interaction that could simultaneously reduce the cognitive and physical costs of *capturing* information while providing natural extensions to easily and optionally *encode* aspects of users' mental context during sensemaking. We hypothesize that such an effective interaction should have the following characteristics:

(1) **Accuracy**: It needs to be accurate and precise enough to lock onto the content the users intend to collect.

(2) **Efficiency**: It should be quick and low-effort to perform, and minimize interruptions to the main activities that users are performing, such as learning and active reading.

(3) **Expressiveness**: It should be extendable to provide natural and intuitive affordances for users to express aspects of their mental context at the moment. In the scope of this work, we would like to have wiggling support encoding valence ratings as well as topic priorities.

(4) **Integration**: It should be a complement to and not interfere with the existing interactions that users already use, such as using the pointer to select text and pictures or click on links.

Below, we present a brief overview of the iterative design exploration leading to the current wiggle-based interactions.

5.3.3 Iterative Design Exploration

To begin our exploration, we took a desktop-first approach and brainstormed various interactions that would address these four design goals. To ground our explorations, we also prototyped these candidate interactions using JavaScript in a browser, which is where a large portion of the reading and collecting happens [72, 145]. Like previous approaches, collecting the desired content, including text and/or images, can be broken down into two main phases: *(a) identifying the desired target* and *(b) triggering the collection.*

One of the interactions we first explored was simply clicking on the desired content (or in the gutter to the left or right) to capture it into the system, similar to existing interactions supported by some text editors such as Microsoft Word. Although straightforward, this interferes

with existing selection methods, and would require users to first enter a "grabber" mode, possibly through a special hotkey combination, which violates both design goals (2) and (4). Next, we experimented with hovering the pointer over the target content and keeping it still for a period of time in order to trigger a collection. This has the benefit of not interfering with existing interaction methods as there is no clicking required, satisfying goal (4). However, research has shown that when heavily engaged in active reading and sensemaking tasks, people often need to select and save information frequently within short time intervals [72, 357], and waiting for a noticeable amount of time will add an inherent cost to every collection operation a user wants to perform and therefore is likely to interrupt the user's main activity, violating design goal (2).

Next, we experimented with using non-click gestures (satisfying goal (4)) performed on the desired target to trigger the selection, since gestures are considered intuitive to perform and widely used in both commercial and academic systems [205, 210, 218, 306, 346]. One of the promising ideas was to use the mouse pointer to sketch out a certain shape over the desired target to trigger a collection. In addition, by varying the shape, it could theoretically support encoding different aspects of users' mental model, such as sketching a "+" for marking it as a "pro" and "-" as a "con" [366], supporting design goal (3). However, similar to using keyboard shortcuts, it is hard for users to learn and memorize the different shapes without special affordances [210, 381]. Furthermore, making sure one sketches out the correct gesture may require non-trivial physical as well as cognitive demand, violating goal (2), and even so, these shapes can have a high false recognition rate, violating goal (1).

We then experimented with gestures that do not require special training or practice in order to perform accurately. One that worked particularly well is wiggling the mouse pointer, i.e., making small ballistic back-and-forth movements, on top of the desired collection target (Figure 5.1-a). Here, the choice of a target could be determined from the average or starting location of the mouse pointer during the gesture, and the user continues to perform the same back-and-forth motion until reaching a certain threshold to trigger the collection. Indeed, prior work has suggested that people naturally use the mouse pointer to guide their attention while reading [159], or even unconsciously have the pointer follow their eye gaze [169], so the pointer could be readily available to initiate a wiggle in place.

This has some additional benefits, such as it seemed natural and intuitive like scratching off something [305, 365], it can be activated without clicking, which can be both cognitively and physically costly [193], and is robust against false positives since only a very specific motion pattern could trigger a collection. Furthermore, it can be chained with optional operations such as swiping in different directions that not only are consistent with the wiggling gesture itself but

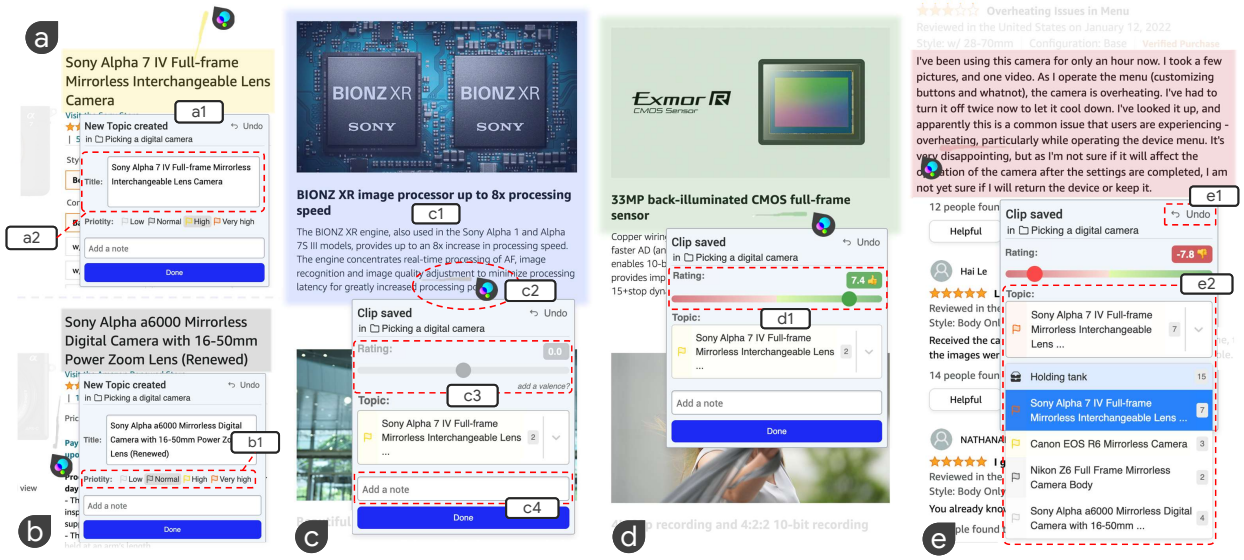


Figure 5.4: Using wiggling to collect information as well as encode priorities and valence ratings. Specifically, as shown in (c), users can wiggle (c2) over the desired content (c1) to collect it into the information holding tank (Figure 5.5-c). A popup dialog will be presented near the just collected content to allow users to optionally add a valence rating (c3), pick a topic that the content should go into (e2), add notes (c4), as well as undo the collection (e1). In addition to regular collection, users can also end the wiggle with a swipe right to encode a positive rating (d) or left to encode a negative rating (e), which can also be changed in the popup dialog (d1). Furthermore, by ending a wiggle with a swipe up (a) or down (b), users can create a new topic with different priorities (b1), and can change the title of the topic directly in the popup dialog (a1).

also intuitively map to users' mental context (such as swiping left/right for negative/positive and up/down for various levels of importance, and even leveraging the amount of distance traveled of a swipe to encode a continuous value).

Since there is no mouse pointer on mobile devices such as smartphones, and using fingers to move left and right in browsers triggers page navigation back and forth, whereas up and down is used for scrolling, we decided to take advantage of these small up-and-down scroll events, since they are not currently in use by any existing interactions. Therefore the wiggling counterpart on mobile devices became using the finger to quickly scroll up-and-down while the finger is over the desired collection target (Figure 5.1-d).

5.4 The Wigglite System

5.4.1 Wiggle-based Gestures

For desktop computers with a traditional mouse, trackpad or trackball input device, the wiggle interaction consists of the following stages, as illustrated in Figure 5.1-a,-b,-c:

(1) **Acquiring the collection target**: To initiate, users move their mouse pointer onto the target content that they would like to collect (Figure 5.1-a0) and initiate the wiggling movement specified in the steps below. Wigglite uses an always-on wiggle gesture recognizer to automatically detect the start of a wiggling gesture. This avoids the requirement of an explicit signal like a keyboard key or mouse down event, which might conflict with other actions, and has the benefit of combining activating and performing the gesture together into a single step, therefore reducing the starting cost of using the interaction technique.

(2) **Wiggle**: To collect the target content, users simply move the mouse pointer left and right approximately inside the target content. To indicate that the system is looking to detect the wiggling gesture, it will display a small "tail" (e.g., Figure 5.4-c2) that follows the pointer on the screen, and replaces the regular pointer with a special one containing the Skeema icon. Wigglite also adds a dotted blue border to the target content to provide feedback about what content will be collected, and the blue color grows in shade as users perform more lateral mouse movements (Figure 5.1-a1–4). This is analogous to half-pressing the shutter button to engage the auto-focus system to lock onto a subject when taking photos with a camera. To assist with collecting fine grain targets, ranging from a word to a block (e.g., a paragraph, an image), Wigglite allows users to vary the average size of their wiggling to indicate the target that they would like to collect: if the average size of the last five lateral movements of a pointer is less than 65 pixels (a threshold empirically tuned that worked well in our pilot testing and user study, but implemented as a customizable parameter that individuals can tune based on their situations), Wigglite will select the word that is covered at the center of the wiggling paths; while larger lateral movements will select a block-level content (details discussed in section 5.4.3.2). In addition, users can abort the collection process by simply stopping wiggling the mouse pointer before there are sufficient back-and-forth movements.

(3) **Collection**: As soon as users make at least five back and forth motions (optimized for the amount of physical effort required and the number of false positive detection through pilot testing, but is also implemented as a parameter that can be customized by individuals in practice, details discussed in section 5.4.3.1), the system will commit to the collection, and gives the target a darker blue background showing that a wiggle has been successfully activated (as shown in Figure 5.1-a5). If users want to collect multiple blocks of content, they can just naturally continue to wiggle over other desired content after this activation. Or, they can stop wiggling. However, if users have selected the wrong target, an undo button appears, which can be clicked to cancel the collection (Figure 5.4-e1).

(4) **Extension**: Instead of just stopping the wiggle motion after collection, users can leverage the last wiggle movement and turn it into a "swipe", either horizontally to the right or left to encode a positive or negative valence rating (as shown in Figure 5.1-b1,b2), or vertically down or up to specify a topic and priority for that topic (as shown in Figure 5.1-c1–4). Feedback for the extension uses different colors for the background of the target content to provide visual salience (details discussed in section 5.4.2).

Similarly, on a mobile device with touch screens:

(1) **Acquiring collection target**: To initiate, a user's finger touches the target content that should be selected.

(2) **Wiggle**: To collect the target block, the user keeps the finger on the screen and starts making small up-and-down scrolling movements. Similar to the desktop scenario, the system adds a dotted blue border to the target content to provide feedback that the wiggling is being detected (Figure 5.1-d0–4). Note that due to the limitations of the large size of the finger with respect to an individual word [72] as well as the unique use cases of mobile devices (e.g., quickly consuming and collecting blocks of information on the go [173, 359]), Wigglite for mobile only supports selecting block-level content such as paragraphs or images.

(3) **Collection**: As soon as the user makes at least five up-and-down motions, the system will commit to the collection by giving the target a darker blue background (Figure 5.1-d5). Now, the user can stop wiggling and lift the finger from the screen. Similar to the desktop version, an undo button pops up that lets the user cancel the collection in case of an error. Note that due to the limited screen real estate that typical mobile devices afford, additional blocks of content will have to be first scrolled into view for users to then capture them, which would make the interaction less fluid. Therefore, collecting multiple blocks of content is currently not supported by Wigglite on mobile.

(4) **Extension**: Instead of stopping the wiggle motion after collection, users can end the wiggle with a horizontal swipe to the left or right to achieve similar encoding capabilities described for the desktop version. After the system detects the wiggle, it turns off other actions until the finger is lifted, so the swipes do not perform their normal actions. (But the normal swipes, scrolling, and other interactions still work normally when not preceded by a wiggle.) Currently, since Wigglite already uses the vertical dimension for detecting wiggling movement on a mobile device, and large cross-screen vertical movements are difficult to perform, especially when holding and interacting with a single hand, we opted not to make a mobile equivalent of encoding topic priorities.

5.4.2 An Overview of The Wigglite System

Wigglite enables users to collect and triage web content via wiggling. First of all, after a regular wiggle with no extension (Figure 5.4-c), Wigglite presents a popup dialog (augmenting the original Skeema popup) directly near the collected content to indicate success. In addition to Skeema's notes field (Figure 5.4-c4), users can attach a valence rating (Figure 5.4-c3) and pick the topic that this piece of information should be organized in (Figure 5.4-e2), as opposed to post-hoc organization using drag and drop as required by Skeema. By default, it goes into the last topic the user picked or the holding tank if none was picked initially. Unlike Skeema where information is saved in pure text format or an inflexible screenshot with limited resolution, Wigglite leverages the technique introduced in Unakite (see Chapter 3, section 3.3.2.2) to preserve and subsequently show the content with its original CSS styling, including the rich, interactive multimedia objects supported by HTML, like links and images. This makes the content more understandable and useful, and also helps users quickly recognize a particular piece of information among many others by its appearance [232].

Of course, a more fluid way to encode user judgments than what was described above is to leverage the natural extension of the wiggle gesture discussed in the previous section: to encode a valence rating in addition to collecting a piece of content, users can end a wiggle with a horizontal "swipe", either to the right to indicate positive rating (or "pro", characterized by a green-ish color that the background of the target content turns into, and a thumbs-up icon, as shown in Figure 5.4-d), or the left for negative rating (or "con", characterized by a red-ish color that the background of the collected block turns into, and a thumbs-down icon, as shown in Figure 5.4-e). Optionally, users can also turn on real-time visualizations of "how much" they swiped to the left or right to encode a rating score representing the degree of positivity or negativity, and can adjust that value in the popup dialog (Figure 5.4-d1) or from the information card (Figure 5.2-f). Under the hood, Wigglite calculates this score as the horizontal distance the pointer traveled leftward or rightward from the average wiggle center divided by the available distance the pointer could theoretically travel until it reaches either edge of the browser window. This score is then scaled to be in the range of -10 to 10 to match with the existing values provided by Skeema.

Alternatively, to directly create a topic and encode it with a priority from wiggling, users can either end the wiggle with a swipe up (encoding "high", characterized by a yellow-ish color that the background of the target content turns into, as shown in Figure 5.4-a) or down (encoding "normal", characterized by a gray-ish color that the background of the target content turns into, as shown in Figure 5.4-b). Optionally, if the user swipes all the way up or down to the edge of the browser window, Wigglite will additionally encode two more levels of priorities, "urgent" and

"low", indicated by a bright orange and a muted gray color (Figure 5.4-b1), which can be adjusted in the popup dialog (Figure 5.4-b1) as well as in the topics view (Figure 5.2-c). In this case, the content will instead be used as the default *title* of the newly created topic (which users can change in the popup dialog directly as shown in Figure 5.4-a2 or later in the topics view).

To help users better manage the information that they have gathered in the holding tank, Wigglite offers several additional features on top of the original Skeema system. First, it enables users to sort the information cards by various criteria, such as in the order of valence ratings or in temporal order (Figure 5.5-b). Second, it offers category filters (Figure 5.5-a) automatically generated based on the encodings that users provided using wiggling (or edited later) and the provenance of information (where it was captured from). Users can quickly toggle those on or off to filter the collected information. For example, in Figure 5.5-b, the information with a "positive rating" or "negative rating" and collected from "amazon.com" was filtered and shown, as indicated by the dark gray background of the corresponding filters (if none of the filters are enabled, all the information cards will be shown). Third, users can quickly filter out information with a lower rating (e.g., indicating that it was less impactful to a user's overall goal and decision making) by adjusting the threshold using the "Focus on clips with a rating over `threshold`" slider shown in Figure 5.2-f. As a result, clips with rating scores lower than the set threshold would be automatically grouped together at the end and grayed out (Figure 5.2-d), and users can easily archive or put them into the trash in a batch by clicking the "Move these clips to trash" button (Figure 5.2-e). These organizational features further help users reduce clutter in the holding tank, and provide a scaffold for them to start dragging and dropping clips into their respective topics.

Due to the limited screen size and use cases of a mobile device, we chose to only let users view the clips along with their valence in the holding tank (Figure 5.5-c).

5.4.3 Design and Implementation Considerations

Here, we discuss important design and implementation considerations made through prototyping Wigglite with JavaScript in a browser to achieve the design goals specified in section 5.3.2.

5.4.3.1 Recognizing a wiggle gesture

For accurately recognizing the wiggle pattern, we explored several options. One way is to use an off-the-shelf gesture recognizer such as the $1 [366] or the Protractor [226] recognizer. Although these recognizers may be lightweight and easy to customize, they are fundamentally designed to recognize distinguishable shapes such as circles, arrows, or stars, while the path of our wiggle gesture does not conform to a particular shape that is easily recognizable (and we argue that it

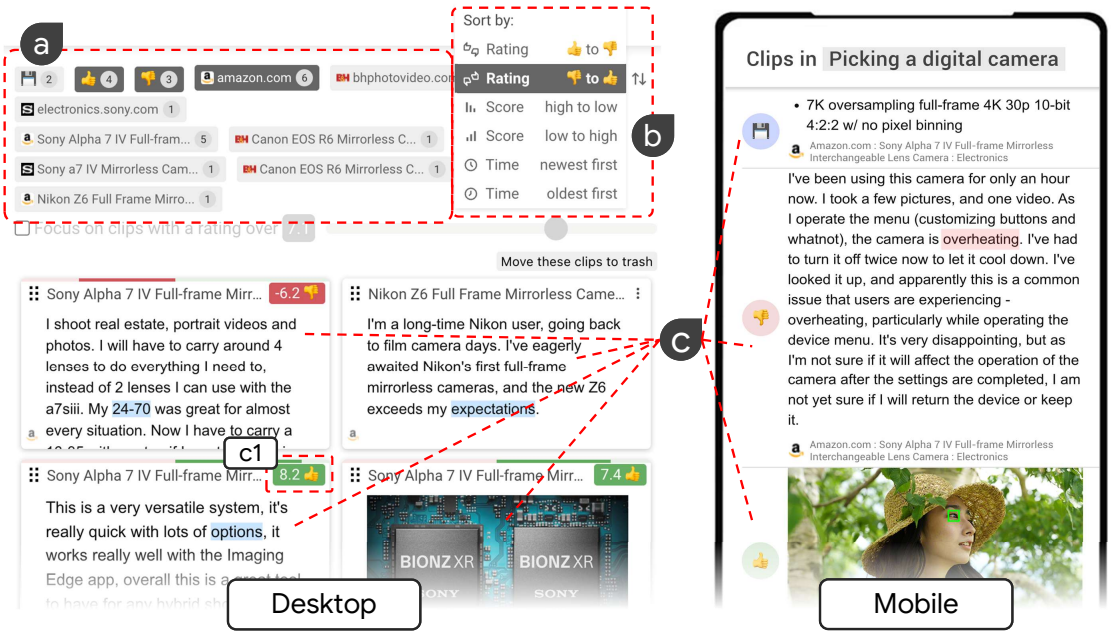


Figure 5.5: Wigglite's information holding tank shown both on desktop and on mobile, which houses content that users collected through wiggling in the form of information cards (c). In addition, on desktop, users can apply different filters (a) and sorting mechanisms (b) to the information cards.

should not conform to any particular shape, the sketching of which would increase the cognitive and physical demand). A second option we investigated was to build a custom computer vision based wiggle recognizer using transfer learning from lightweight image classification models such as MobileNets [165]. Though these ML-based models improved the recognition accuracy in our internal testing, they incurred a noticeable amount of delay due to browser resource limitations (and limitations in network communication speed when hosted remotely). This made it difficult for the system to perform eager recognition [306] (recognizing the gesture as soon as it is unambiguous rather than waiting for the mouse to stop moving), which is needed to provide real-time feedback to the user on their progress.

To address these issues, we discovered that a common pattern in all of the wiggle paths that users generated with a mouse or trackpad during pilot testing share the characteristic that there were at least five (hence the activation threshold mentioned in section 5.4.1) distinguishable back and forth motions in the horizontal direction, but inconsistent vertical direction movements. Similarly, on smartphones, wiggling using a finger triggers at least five consecutive up and down scroll movements in the vertical direction but inconsistent horizontal direction movements. Therefore, we hypothesized that only leveraging motion data in the principle dimension (horizontal on desktop, and vertical on mobile) would be sufficient for a custom-built recognizer to differentiate intentional wiggles from other kinds of motions by a cursor or finger.

Based on our implementation using JavaScript in the browser, we found that it successfully supports real-time eager recognition with no noticeable impact on any other activities that a user performs in a browser. Specifically, the system starts logging all mouse movement coordinates (or scroll movement coordinates on mobile devices) as soon as any mouse (or scroll) movement is detected, but still passes the movement events through to the rest of the DOM tree elements so that regular behavior would still work in case there is no wiggle. In the meantime, the system checks to see if the number of reversal of directions in the movement data in the principle direction exceeds the activation threshold, in which case a "wiggle" will be registered by the system. After activation, the system will additionally look for a possible subsequent wide horizontal or vertical swipe movement (for creating topics with priority or encoding valence to the collected information) without passing those events through to avoid unintentional interactions with other UI elements on the screen. As soon as the mouse stops moving, or the user aborts the wiggle motion before reaching the activation threshold, the system will clear the tracking data to prepare for the next possible wiggle event.

5.4.3.2 Target Acquisition

In order to correctly lock onto the desired content without ambiguity, we explored two approaches that we applied in concert in Wigglite. The first approach is to constrain the system to only be able to select certain targets that are usually large enough to contain a wiggling path and are semantically complete. For example, one could limit the system to only engage wiggle collections on block-level semantic elements [1], such as `<div>`, `<p>`, `<h1>-<h6>`, `<li>`, `<img>`, `<table>`, etc. This way, the system will ignore inline elements that are usually nested within or between a block-level element. This approach, though sufficient in a prototype application, does rely on website authors to organize content with semantically appropriate HTML tags.

The second approach is to introduce a lightweight disambiguation algorithm that detects the target from the mouse pointer's motion data in case the previous one did not work, especially for a small `<span>` or an individual word. To achieve this, we chose to take advantage of the pointer path coordinates (both X and Y) in the last five lateral mouse pointer movements, and choose the target content covered by the most points on the path. Specifically, we used the same re-sampling and linear interpolation technique introduced in the $1 gesture recognizer [366] to sample the points on a wiggle path to mitigate variances caused by different pointer movement speeds as well as the frequency at which a browser dispatches mouse movement events. On mobile devices, since the vertical wiggling gesture triggers the browser's scrolling events, the

target moves with and stays underneath the finger at all times. Therefore, we simply find the target under the initial touch position.

When Wigglite is unable to find a target (e.g., when there is no HTML element underneath where the mouse pointer or the finger resides) using the methods described above, it does not trigger a wiggle activation (and also not the aforementioned set of visualizations), even if a "wiggle action" was detected. This was an intentional design choice to further avoid false positives as well as to minimize the chances of causing distractions to the user.

5.4.3.3 Integration with existing interactions

Notice that the wiggling interaction does not interfere with common active reading interactions, such as moving the mouse pointer around to guide attention, regular vertical scrolling or horizontal swiping (which are mapped to backward and forward actions in both Android and iOS browsers) [261, 337]. In addition, wiggling can co-exist with conventional precise content selection that are initiated with mouse clicks or press-and-drag-and-release on desktops or long taps or edge taps on mobile devices [78, 303]. Furthermore, unlike prior work that leverages pressure-sensitive touch screens to activate a special selection mode [72], wiggling does not require special hardware support, and can work with any kind of pointing device or touch screen.

5.4.4 Implementation Notes

We implemented the wiggling technique as an event-driven Java-Script library that can be easily integrated into any website and browser extension. Once imported, the library will dispatch `wiggle`-related events once it detects them. Developers can then subscribe to these events in the applications that they are developing. All the styles mentioned above are designed to be easily adjusted through predefined CSS classes. The library itself is written in approximately 1,100 lines of JavaScript and TypeScript code.

The Wigglite browser extension is implemented in HTML, TypeScript, and CSS and uses the React JavaScript library [109] for building UI components. It uses Google Firebase for backend functions, database, and user authentication. In addition, the extension is implemented using the now standardized Web Extensions APIs [263] so that it would work on all major browsers, including Google Chrome, Microsoft Edge, Mozilla Firefox, Apple Safari, etc. However, we primarily targeted Google Chrome and Microsoft Edge to minimize testing efforts during development.

The Wigglite mobile application is implemented using the Angular JavaScript library [134], the Ionic Framework [172] and works on both iOS and Android operating systems. Due to the limitations that none of the current major mobile browsers have the necessary support for de-

veloping extensions, Wigglite implements its own browser using the `InAppBrowser` plugin from the open-source Apache Cordova platform [35] to inject into webpages the JavaScript library that implements wiggling as well as custom JavaScript code for logging and communicating with the Firebase backend.

5.5 Performance Evaluation

We carried out a study to evaluate the *speed* and *accuracy* of the wiggling gesture when performing selections on blocks of content. The study was carried out separately on desktop computers and smartphones.

Participants for this study were recruited from Amazon Mechanical Turk who had more than 1000 accepted tasks with above 95% acceptance rate and lived in countries that primarily spoke English.

In order to track user performance, we instrumented the aforementioned JavaScript library with various tracking capabilities, including keeping track of key timestamps such as the start and end of a wiggle gesture, and all the cursor or touch movements in between. In addition, we built similar tracking capabilities into our implementations of the baseline (control) conditions.

5.5.1 Procedure

We recruited 60 participants (age 19-69; Mean=37.3; SD=10.4; 40 males and 20 females) for the desktop version of the study. Each participant was compensated 3.50 US dollars for an average of 21.2 minutes. For the smartphone version of the study, we recruited a total of 64 participants (age 18-69; Mean=34.3; SD=10.9; 39 males and 25 females). Each of these participants was also compensated 3.50 US dollars, and they took an average of 21.4 minutes. All smartphone participants were required to have and normally use an Apple iOS iPhone to ensure font and layout consistency. Each version included both a wiggle and a conventional selection group of tasks on their device.

Before the study began, participants signed a consent form and filled out a pre-survey about their demographic information as well as the type of pointing device if desktop (a regular mouse with a scroll wheel, a touchpad, or a trackball), and the make and model of the computer or smartphone that they would use to perform the study. We also collected information about which hand they would use to operate the pointing device or interact with their smartphone screen and asked them not to switch hands during the study. At the beginning of the study, the participants were instructed to read through a brief tutorial to learn how the wiggle interaction

works, which included multiple small demonstration videos. The participants then went through a practice session in which they were required to use wiggling to select paragraphs of different sizes, ranging from single characters to several lines. A similar practice session but with conventional selection techniques was administered afterward: for the desktop version of the study, participants were asked to select the paragraphs by moving their mouse cursor to the start of the paragraph, pressing the mouse left button, dragging the cursor to the end of the paragraph, and releasing the mouse left button; for the smartphone version of the study, participants were asked to long press on the first word of a paragraph and then drag the selection handles to include the entire paragraph without lifting the finger. After both practice sessions, the participants were presented with two groups of tasks, where they were asked to complete one of the groups using the wiggling technique, the other with the baseline conventional selection methods. The order of whether conventional or wiggle selection was first was counterbalanced across participants. Each group consists of two tasks, both of which used the assigned selection method to select random paragraphs in a document without actually reading it. In task 1, the paragraphs contained dummy text, with a constant font size (14px), and varied locations and sizes of the 19 selection targets (1 character up to 10 lines — all participants saw identical pages). Participants were instructed to select the target marked by a blue arrow on the left in each step, and would automatically be advanced into the next step once a successful selection is made (as shown in Figure 5.6). In task 2, the content was adapted from a real blog post from `medium.com` on traveling to Hawaii, and we selected 18 HTML elements (including `<h1>`–`<h6>`, `<p>`, `<li>`) as the targets. Both the condition and document orders were counterbalanced. After each condition, participants filled out a NASA TLX survey for that condition to measure the cognitive load.

The test app that the participants used was written using React [109]. The documents that participants performed selections on were embedded in an `iframe` component isolated from the main application where the tutorials and task instructions reside. The `iframe` was specified to be 700 by 700 pixels on desktop browser, and 290 by 360 pixels in the mobile condition. The `iframe` used the `Window.postMessage()` API to communicate tracking data to the host app, and the host app uploaded the data to a Google Firestore database in real-time.

5.5.2 Results

To examine how wiggling performs compared to conventional selection methods, we first compared the overall duration and accuracy for participants to finish the entire set of selection tasks. Since participants must select each target correctly before moving on to the next target, a user's

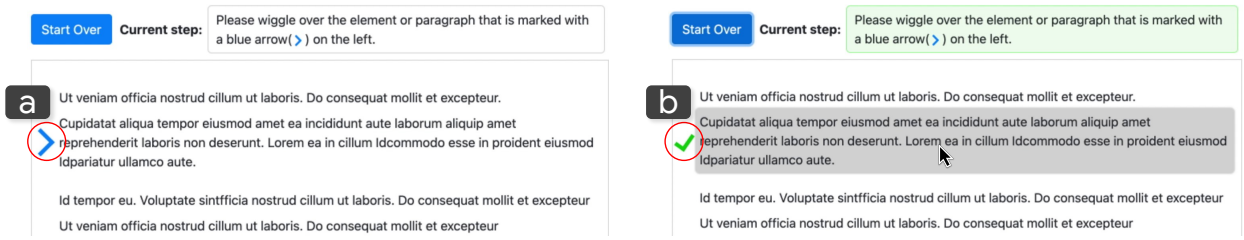


Figure 5.6: Setup of the wiggling performance evaluation conducted on Amazon Mechanical Turk. After a participant successfully selects the correct target marked by a blue arrow on the left (a) in the current step, a green check mark will show up next to the target (b) and the participant will automatically be advanced into the next step. If the user selects incorrectly, this is counted as an error, and the instruction at the top and the blue arrow animate briefly, and the user tries this step again.

overall selection accuracy is defined as the total number of targets divided by the total selection operations attempted.

Results show that using wiggling to select was faster and more accurate than using the conventional selection methods. On desktop (shown in Figure 5.7 (a) & (b)), participants completed both tasks significantly faster using the wiggling interaction (for task 1: 45.5 vs 72.7 seconds, $p<0.001$; for task 2: 36.4s vs 52.5s, $p=<0.001$; based on paired T-tests). Additionally, participants completed both tasks significantly more accurately using the wiggle interaction (for task 1: 91.0% vs 79.3% , $p<0.001$; for task 2: 97.2% vs 89.9%, $p=<0.001$).

This performance difference between the two was even more pronounced on mobile devices (shown in Figure 5.8 (a) & (b)), participants completed both tasks significantly faster (for task 1: 62.8 vs 99.1 seconds, $p<0.001$; for task 2: 45.2 vs 84.1 seconds, $p<0.001$), and were more than twice as accurate (for task 1: 99.7% vs 45.1%, $p<0.001$; for task 2: 98.4% vs 43.9%, $p=6.9e-49<0.001$; based on paired T-tests).

One explanation for the platform-independent difference between the two methods is that the additional opportunity for error in block selection afforded by the precision of the baseline selection method. Although users in both conditions were given the same selection tasks, users in the wiggle condition could only get the selection incorrect by wiggling and activating the selection on the wrong target. In contrast, users in the baseline condition could additionally not fully include the target content (under-select, perhaps from lifting their finger or mouse button too early), or include some extra content (over-select). Indeed, a large percentage of errors (97% in mobile, 92% in desktop) in the baseline condition were under or over-selections. To avoid either of these selection errors, users likely spent additional time ensuring a precise starting and ending point for their selection.

Even though we did not specifically design the wiggling interaction for small selections, we still included them as targets in the study to understand the boundary conditions. Figure 5.9

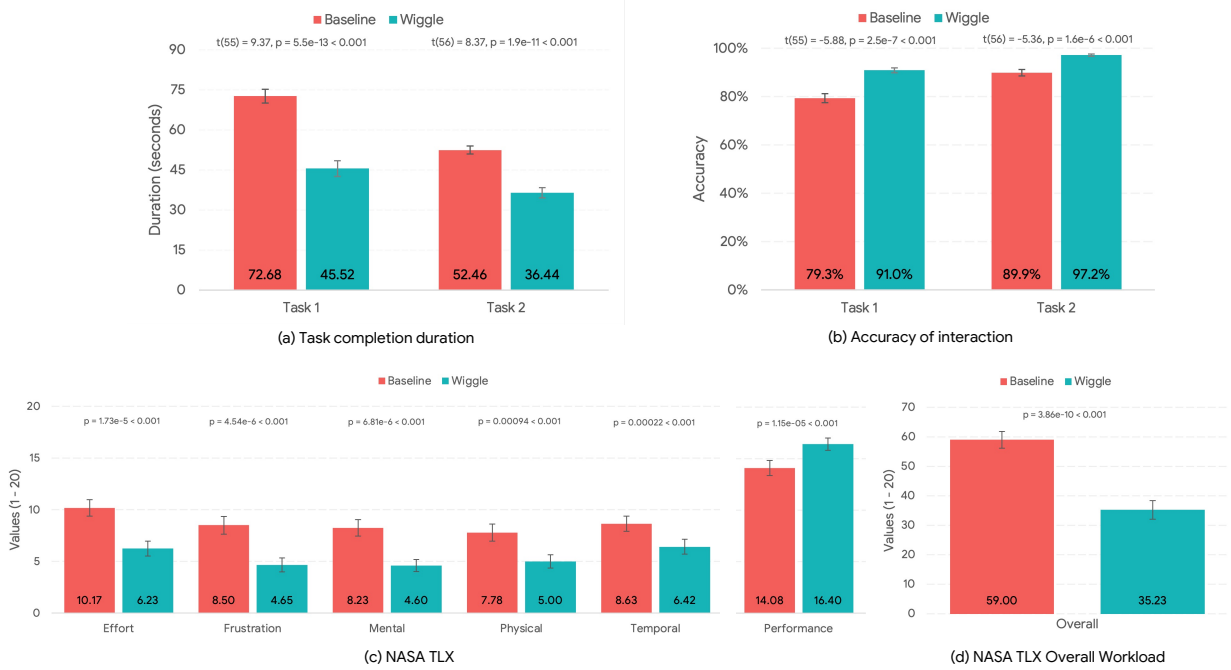


Figure 5.7: Desktop Results: Duration (a) and accuracy (b) of wiggling compared to conventional selection. Using wiggling was significantly faster and more accurate. (c, d) In addition, participants also had lower perceived workload but higher perceived performance based on the NASA TLX survey. All differences between conditions are statistically significant.

shows the duration of selection on desktops broken down by target sizes from as small as a single character to as large as 10 lines. On desktop computers, for targets that are more than 25 characters in width all the way up to 10 lines, using wiggling to select is significantly faster than using the conventional selection method. As shown in Figure 5.9, the speed of completion *decreases* with the size of the selection targets for wiggling, but *increases* for conventional selection. We hypothesize that for smaller-sized targets, the participants were careful to ensure that the majority of the wiggling path was within the vicinity of a target, hence the slowdown. In addition, one-way ANOVA result suggests that there is no statistically significant difference between the completion time when using wiggling to select 2-line, 3-line, 5-line, and 10-line targets ($F(3,444)=0.129$, $p=0.943$), suggesting that the wiggling speed has plateaued as the target size gets larger, and participants were able to wiggle inside these larger blocks comfortably. Based on these trends, it is reasonable to believe that wiggling would also perform better in terms of speed for targets that are more than 10 lines in height. The apparent increase in selection time for conventional selection might be because it required longer movements from the start to the end of the blocks, which is supported by prior work [72, 78, 303].

Figure 5.10 shows the duration of selection on smartphones broken down by target sizes from as small as a single character to as large as 10 lines. On smartphones, for targets that are one

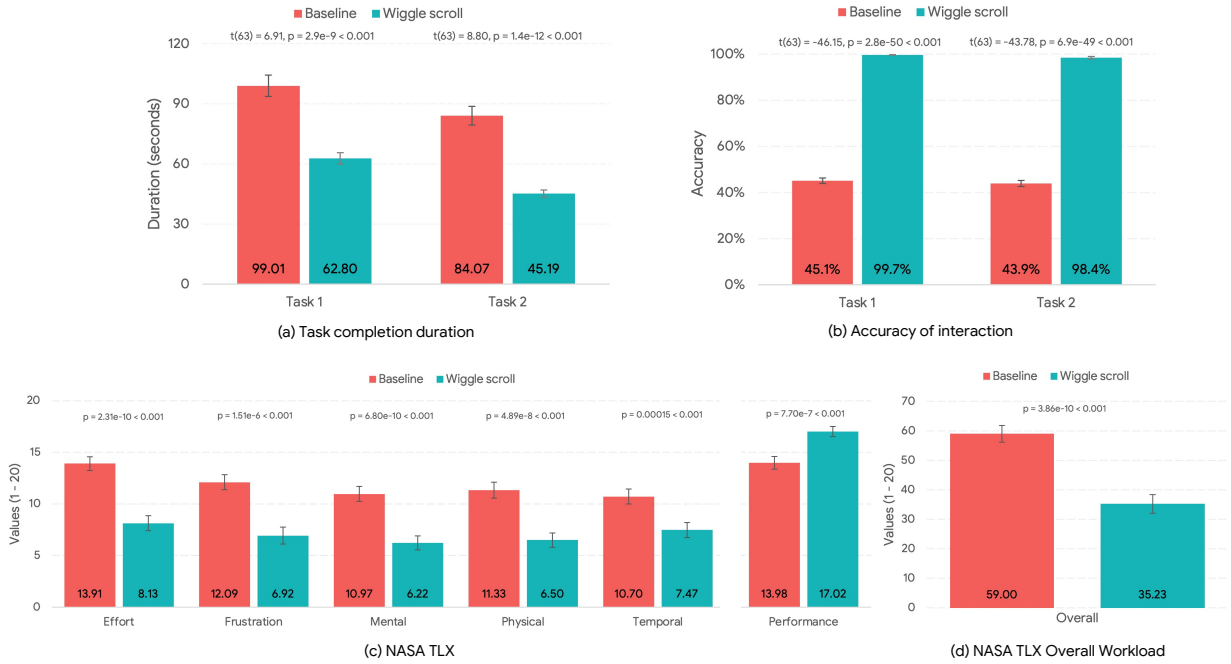


Figure 5.8: Smartphone Results: Duration (a) and accuracy (b) of wiggling compared to conventional selection. Using wiggling was significantly faster and more accurate. (c, d) In addition, participants also had lower perceived workload but higher perceived performance based on the NASA TLX survey. All differences between conditions are statistically significant.

character in size (approximately 14 by 14 pixels), the speed of selection using wiggling is still faster than using the conventional method, however, the difference is not statistically significant. We hypothesize that when trying to locate targets of such small sizes, participants suffer from the same touch inaccuracy problem as they would when using long press and selection handles to select [72], especially when their finger would be occluding the actual target. For targets that are more than 2 characters in width all the way up to 10 lines, using wiggling to select is significantly faster than using the conventional selection method, which is consistent with the overall speed comparison shown previously, suggesting that using wiggling can mitigate the touch inaccuracy problem and therefore improves the speed of block selections. An additional one-way ANOVA test suggests that there is no statistical significant difference between the completion time when using wiggling to select targets that are more than 2 characters in size on smartphones ($F(9,1270)=1.31$, $p=0.224$), suggesting that the wiggling speed has plateaued and remains essentially constant regardless of the size of the targets. Therefore, based on these trends, it is reasonable to believe that using wiggling to select targets that are larger than 10 lines in size would also be faster.

Although we specifically asked the participants to wiggle five or more times in the instructions, we were still interested in cases where they tried to trigger a selection with wiggling but

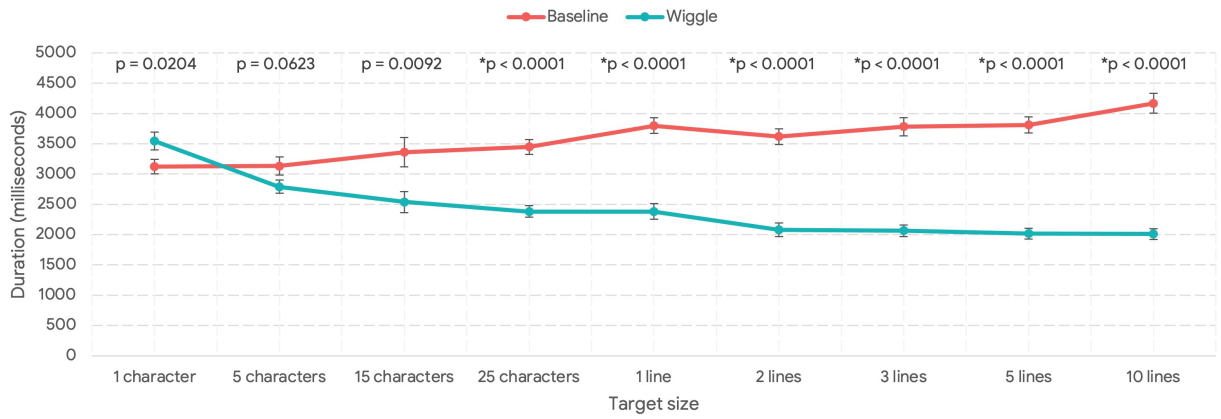


Figure 5.9: Duration of completion on **desktops** broken down by the size of the selection targets. Nine paired T-tests were performed for each target size. Asterisks (*) indicate statistically significant results, corrected for multiple comparisons.

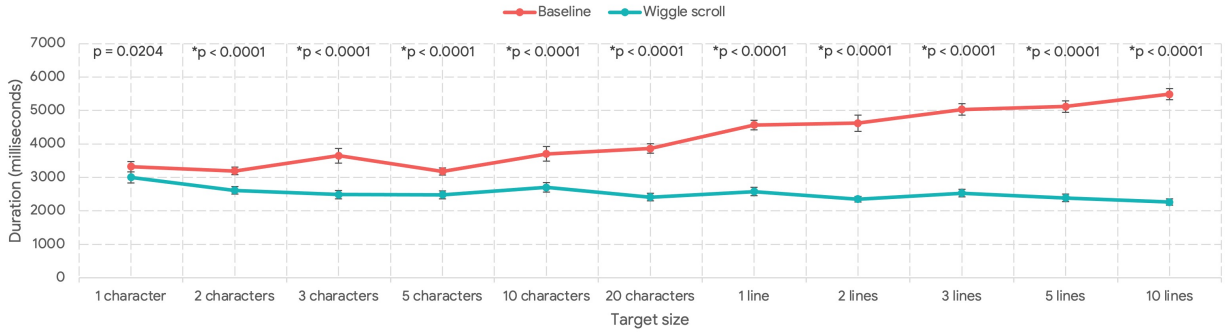


Figure 5.10: Completion time on **smartphones** broken down by the size of the selection targets. Asterisks (*) indicate statistically significant results, corrected for multiple comparisons.

failed since they wiggled less than five times, i.e., false negatives. Here, we report the false negative rates, defined as the number of false negatives divided by the total number of would-be wiggle activations if we set the activation threshold to be 4 instead of 5. On desktop, the false negative rate is 0.76% (SD=2.35%), while on smartphones, the false negative rate is 0.07% (SD=0.54%). This suggests that the majority of times participants were able to wiggle 5 or more times, confirming our choice of this threshold.

In addition, we were also interested in seeing the sensitivity of the current wiggling implementation. Therefore, we had the wiggling engine turned on in the background when participants were doing baseline tasks, and looked at how many times a wiggle would have been activated on any paragraphs, i.e., false positives. For this analysis, we excluded the wiggles that were triggered before any conventional selections were attempted on the first target in each task since participants may intentionally try to wiggle to see if it would work at the beginning. On desktops, there were 5 accidental triggers across 117.3 minutes of doing conventional selection tasks with mouse

moving around in the document, suggesting that the chance of registering false positive wiggles was very low. By examining the mouse movement traces of these false positives, we uncovered that they came from two particular participants who seemed to be repeatedly trying to wiggle and see if it would work even after using conventional selection to select the first target. On smartphones, there was only 1 false positive across all tasks and participants, and by examining the corresponding scrolling traces, it appears that the participant was rapidly scrolling up and down to find the next selection target marked with a blue arrow, so it is an actual false positive. These results suggest that using five times as the wiggling activation threshold is indeed effective towards preventing false positives.

Participants also perceived the process to have lowered the workload when using wiggling based on their perceived effort, frustration, mental, physical, and temporal demands based on the NASA TLX survey (shown in Figure 5.7 and Figure 5.8 (c), (d)) as well as perceived increased in performance on both desktop and smartphones. This suggests that wiggling, despite its relative novelty, can significantly reduce the current interaction costs of selecting blocks of content.

In our post-completion survey, many participants across both desktop and mobile found the new wiggling interaction a fresh take on the selection interaction that they perform frequently. Below are representative quotes from participants:

> "This [wiggling on desktop] is remarkable. I want it like yesterday. Seriously though - this is an amazing take on an old and irritating thing that we all have to do fairly often. Brilliant."

> "The wiggle scrolling would have been nice to have when I was working on my car and needed all the specific fastening torque numbers and needed to collect them quick and efficiently while reading through the information and to not break my concentration on what I was reading."

> "[if it were implemented]... it will save quite a bit of time for me to highlight contents which can be frustrating sometimes because I often miss highlighting one letter at the beginning or at the end of the text. This wiggle scrolling is very innovative and useful in minimizing frustration."

5.6 User Study

In addition, we conducted a lab study to evaluate the usability and usefulness of Wigglite in helping people collect information as well as encode aspects of their mental context while doing so. Specifically, we aimed to address the following research questions:

- **RQ1 [Accuracy]**: Are wiggle-based interactions sufficiently accurate to help users collect what they want?
- **RQ2 [Efficiency]**: Are wiggle-based interactions sufficiently low-friction to perform without interrupting the primary reading and sensemaking activities?
- **RQ3 [Expressiveness]**: Are the proposed extensions of marking priorities and valence useful in helping people encode their mental contexts?
- **RQ4 [Integration]**: Do wiggle-based interactions interfere with existing interactions that people are already using?

5.6.1 Participants

We recruited 12 participants (6 male, 6 female; 3 students, 3 software engineers, 2 UX designers, 1 UX researcher, 1 medical doctor, 1 administrative staff member, and 1 entrepreneur) aged 21-38 years old (mean age = 28.5, SD = 4.5) through emails and social media. Participants were required to be 18 or older and fluent in English. All participants reported experience reading and making sense of large amounts of information online for either professional or personal purposes on a daily basis, and had tried or were using commercially available web clipping and organization tools and systems, such as the Evernote Clipper, OneNote, or Notion.

5.6.2 Study Methodology

The study was a within-subjects design with each participant engaging in two tasks, one using Wigglite with Skeema in the experimental condition, and the other just using Skeema in the control condition, counterbalanced for order. For our control condition, Skeema provided the affordances of a web clipping tool, which would provide a more conservative and matched baseline than no tool support. Specifically, our control condition enabled participants to capture text through a popup button (Figure 5.3-a1) to save highlighted text and a screenshot clipper instead of the wiggle interaction. After saving the information, participants could set the priority of topics and the valence of information in the workspace view (Figure 5.2), versus being able to encode them as a continuation of the wiggle in Wigglite.

For each task, participants were presented with a product category they needed to research, and a set of three Amazon pages from which they were required to collect information. Participants were instructed to read through the provided webpages, collect information, and organize the information clips into topics, such as by different options or different criteria in which the options should be evaluated. They were required to at least collect 10 information clips as well as

	Condition	Overhead cost	Time (seconds)	*n* of clips collected	*n* of topics created using wiggling	*n* of topics created separately in the workspace view	Total *n* of topics created
Task A	Baseline	33.0% (8.60%)	713.7 (76.0)	21.0 (7.03)	N/A	4.17 (1.17)	4.17 (1.17)
	Wigglite	14.0% (7.89%)	558.7 (76.5)	38.3 (5.28)	7.50 (1.05)	0.50 (0.84)	8.00 (1.89)
Task B	Baseline	30.40% (7.31%)	692.0 (131.4)	19.5 (6.81)	N/A	4.67 (0.52)	4.67 (0.52)
	Wigglite	12.8% (2.74%)	515.7 (54.3)	37.3 (8.64)	7.17 (1.17)	0.50 (1.22)	7.67 (2.39)
Average	Baseline	31.7% (7.73%)	702.8 (102.9)	20.3 (6.68)	N/A	4.42 (0.90)	4.42 (0.90)
	Wigglite	13.4% (5.67%)	536.8 (67.0)	37.8 (6.85)	7.33 (1.07)	0.50 (1.00)	7.83 (2.07)

Table 5.1: Statistics of the performance measures in the study. Standard deviations are included in the parentheses.

create a minimum of 3 topics with priority for each task. Participants had 15 minutes to complete the task, but could inform the experimenter to move on if they finished early.

The two tasks were:

- *(A)* Choosing a digital mirrorless camera: participants were told to imagine that they were to purchase a new mirrorless camera to take photos of their spouse and young kids on their weekend road trips.
- *(B)* Buying a vacuum cleaner: participants were told to imagine that they were to buy a new vacuum cleaner in preparation for moving into a new house with a newborn baby and their two pets.

In order to minimize differences between tasks and participant decision making, we provided a fixed set of web pages per task, each with approximately eight screens of content. As described in the results, the two tasks took approximately the same amount of time for participants to finish, and were counterbalanced in order and randomized across conditions.

Each study session started by obtaining consent and having participants fill out a demographic survey. Participants were then given a 10-minute guided tutorial showcasing the various features of Wigglite as well as the baseline system, and a 10-minute free-form practice session to familiarize themselves with the features of both systems. At the end of the study, participants completed a survey and engaged in a semi-structured interview about their experience with the tool. The interview focused on participants' perceptions of using the wiggle-based interactions. The questions probed the perceived effectiveness of wiggling, their current practices around collecting information, and scenarios where they thought wiggling would be useful and how they would modify it to be more useful. The interviews were audio-recorded and transcribed, after which qualitative coding and thematic analysis [77] were performed.

Each study session took approximately 60 minutes to complete, using a designated MacBook computer with Google Chrome and Wigglite installed as well as a Logitech MX Master 2S mouse.

5.7 Results

All participants were able to complete each task within the specified 15 minute time limit. Below, we compile together both quantitative and qualitative evidence to evaluate Wigglite with respect to our four design goals and research questions.

5.7.1 RQ1 [Accuracy]

First, evaluate if the wiggling gestures are accurate enough to help users collect and express what they want. Specifically, we looked for cases where: (1) participants hit the undo button to dismiss an incorrect wiggle activation and redo the wiggling due to Wigglite picking up the wrong target content, which turned out to be on average 0.67 (SD = 0.65) times per person per task, and only accounted for 1.48% of the 45.16 (SD = 8.82) total wiggle actions participants on average performed per task; (2) participants had to use the popup dialog to immediately edit the valence or topic priority because Wigglite picked the wrong swipe direction, which turned out to be 0; (3) participants had to redo the wiggling gesture because the previous one they performed did not activate at all, which turned out to be on average 0.92 (SD = 0.67) times per person per task, and only accounted for 2.01% of the total wiggle actions participants on average per task.

This evidence suggests that the wiggling technique provided by the current Wigglite system is sufficiently accurate and robust, at least with ample amount of training and practice. It would be interesting for future work to explore how it performs in the wild, potentially without much upfront practice, and examine whether and how people's wiggling accuracy and performance evolve over time.

5.7.2 RQ2 [Efficiency]

Second, we are interested in understanding if Wigglite creates a more fluid experience when collecting and triaging information with less interruption compared to the baseline condition. For this comparison, we opted to measure two key metrics: the *overhead cost* of using a tool to collect and triage information, and the total amount of *time* it took for participants to finish each task. For the Wigglite condition, we calculate the overhead cost as the portion of the time participants spent on directly interacting with Wigglite (performing wiggling gestures, interacting with the popup dialog if necessary, filtering the information clips, organizing them in the workspace view, etc.) out of the total time they used for a task (vs. reading and comprehending the web pages) [232, 234]. Similarly, in the baseline condition, the overhead cost accounts for situations

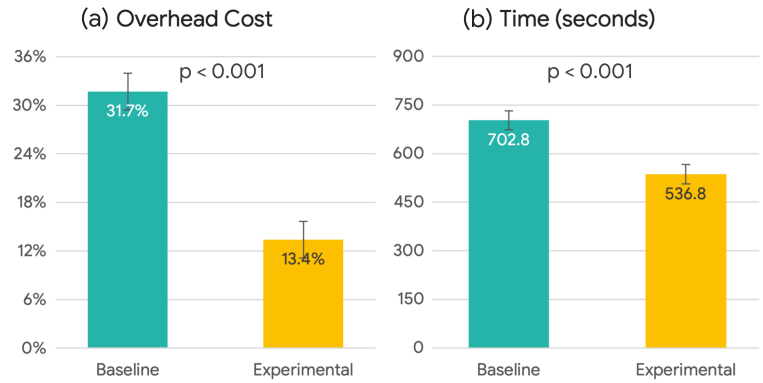


Figure 5.11: Using Wigglite incurred significantly less overhead cost (a) and helped participants finished the tasks significantly faster (b) when compared to the baseline condition in the user study.

where participants use the highlighting or screenshot feature to collect information, organize them in the workspace view, etc.

We conducted a two-way repeated measures ANOVA to examine the within-subject effects of condition (Wigglite vs. baseline) and task (A vs. B) on overhead cost. There was a statistically significant effect of condition ($F(1,20) = 40.7$, $p < 0.001$) such that the overhead cost was significantly lower (58% lower, as shown in Table 5.1 and Figure 5.11-a) in the Wigglite condition (Mean = 13.4%, SD = 0.06) than in the baseline condition (Mean = 31.7%, SD = 0.08). There was no significant effect of task ($F(1,20) = 0.46$, $p = 0.51$)). In addition, a two-way repeated measures ANOVA was conducted to examine the within-subject effects of condition (Wigglite vs. baseline) and task (A vs. B) on task completion time. There was a statistically significant effect of condition ($F(1,20) = 20.8$, $p < 0.001$) such that participants completed tasks significantly faster (23.6% faster, as shown in Table 5.1 and Figure 5.11-b) with Wigglite (Mean = 536.8 seconds, SD = 67.0 seconds) than in the baseline condition (Mean = 702.8 seconds, SD = 102.9 seconds). Again, there was no significant effect of task ($F(1,20) = 0.77$, $p = 0.38$).

As the condition had a statistically significant impact on both the overhead cost as well as the task completion time (with faster completion and lower overhead cost in Wigglite conditions), Wigglite indeed helped participants reduce the overhead costs of collecting and triaging information and speed up their sensemaking process overall, even though the majority of their time was necessarily spent reading and understanding the material in both conditions.

Furthermore, in the post-study interview, participants overall appreciated the increased efficiency afforded by Wigglite, especially using the wiggling gestures. Many (9/12) mentioned that the perceived workload to collect information that they have encountered was minimal, saying that "*It felt like I didn't do anything to get those snippets into the system*" (P3), and was fluid enough that it did not interrupt their flow of reading the task pages, such as "*I just wiggle and move on,*

in fact, when I am wiggling on something, my eyes are already onto the next paragraph, no more stopping to do the regular clipping thing any more" (P11). Therefore, Wigglite did offer a more fluid experience when collecting and rating information with less interruption.

5.7.3 RQ3 [Expressiveness]

Third, we wanted to know to what extent Wigglite induces changes in people's behavior, especially given the natural extension that wiggling affords to encode priorities and valence.

As shown in Table 5.1, participants collected significantly more information using wiggling (on average 37.8 clips, SD = 6.85) than when using the conventional selecting or screenshot workflow (on average 20.3 clips, SD = 6.68) ($p < 0.01$), despite spending less time on the tasks. Among the collected information clips using wiggling, 75.3% of them were encoded with either a positive or negative valence. Similarly, participants created significantly more topics using Wigglite (on average 7.83 topics, SD = 2.07) than in the control condition (on average 4.42 topics, SD = 0.90) ($p < 0.01$), where topics were required to be created separately in the workspace view. It is also worth noting that using wiggling to create topics (7.33 times, SD = 1.07) almost eliminated the need to separately (0.50 times, SD = 1.00) create topics (granted that most participants did at least edit the title of the topics in the popup dialog or in the workspace view to make them more succinct and easier to read). This evidence suggests that participants indeed were able to use Wigglite to externalize the perceived utility of a particular piece of information as well as their mental judgments of how it aligned with their goals in situ.

Furthermore, in the post-study interviews, some (4/12) participants reflected that Wigglite would enable them to express their perceived utility in a way that is also useful for subsequent sorting and ranking. For example, P5 mentioned that "*I really enjoyed the threading [creating topics with priorities] feature, being able to say something is important or extra important on the spot would help me stay on top of my todo list.*" However, perhaps due to the limited scale of the lab study, we did not observe significant differences in the types of information participants used as topics–most of them are about the different options as well as some criteria to evaluate a product. Future and potentially larger-scale investigations are required to understand the types of information users collect using a lightweight gesture like wiggling versus using conventional capturing methods.

Question Statements	Wigglite condition	baseline condition
I would consider my interactions with the tool to be understandable and clear.	6.25 (0.45)	6.17 (0.72)
I would consider it easy for me to learn how to use this tool.	6.42 (0.67)	6.33 (0.49)
I enjoyed the features provided by the tool.	6.25 (0.62)	6.08 (0.67)
Using this tool would help make my information collection and triaging processes more efficient and effective.	6.17 (0.39)*	5.75 (0.62)*
If possible, I would recommend the tool to my friends and colleagues.	6.33 (0.49)*	5.83 (0.39)*

Table 5.2: Statistics of scores in the post-tasks survey. Participants were asked to rate their agreement with statements related to their experience interacting with Wigglite and the baseline on a 7-point Likert scale from "Strongly Disagree" (a score of 1) to "Strongly Agree" (a score of 7). Statistics in column 2 and 3 are presented in the form of mean (standard deviation). Statistically significant differences ($p < 0.05$) through paired t-tests are marked with an *. The survey questions and scales were adapted from a validated SUS scale [213].

5.7.4 RQ4 [Integration]

Last but not least, we explored if the wiggle gesture would interfere with participants' normal behaviors during web browsing, such as unconsciously using the mouse pointer to guide their attention [169], clicking [158], or scrolling (false positives). To measure this, we looked for cases where participants hit the undo button to dismiss a wiggle activation due to Wigglite had wrongfully recognized some regular mouse movements as a wiggle, which turned out to be 0 across the board. This provides evidence that the wiggling gestures added by Wigglite do not interfere with the existing interactions and user behaviors.

5.7.5 Other Subjective Feedback

In the survey, participants reported (in 7-point Likert scales) that they thought the interactions with Wigglite were understandable and clear (Mean = 6.25, SD = 0.45), Wigglite was easy to learn (Mean = 6.42, SD = 0.67), and they enjoyed Wigglite's features (Mean = 6.25, SD = 0.62). In addition, compared to the baseline condition (Mean = 5.75, SD = 0.62), they thought using Wigglite (Mean = 6.17, SD = 0.39) would help make their information collection and triaging processes more efficient and effectively (p = 0.017), and would recommend Wigglite (Mean = 6.33, SD = 0.49) over the baseline version of Wigglite (Mean = 5.92, SD = 0.29) to friends and colleagues (p = 0.007), both differences were statistically significant under paired t-tests. Details of the survey questions and scores are presented in Table 5.2.

In addition, some participants reflected on the playfulness and attractiveness of the wiggle interactions and how it encouraged them to collect information compared to what they normally have to go through. For example, P8 said: "*It's fun, you know? I didn't quite believe it at the beginning, but it actually made grabbing stuff so much fun*", and P1 suggested that "*somehow with*

this, I don't think going through something that I'm not familiar with would be as daunting as it used to be". Four of the participants even went on to ask when Wigglite will be released publicly so that they could use it for their own work and personal tasks, and wondered if they could customize the system, such as by "*writing some sort of plugin, like the one I wrote for Obsidian [271], to map the different directional swipes to what I want depending on the situations that I'm in*" (P11).

5.8 Limitations

One potential limitation to wiggling is the suitability of its rapid back-and-forth movements to user populations with motor impairments or advanced age, for example, users with hand tremors. There are several ways in which wiggling might be more suitable than expected or relatively easily adapted to such populations. First, since wiggling uses the initial mouse location as its selection anchor, a user can take their time adjusting to arrive at the correct area (which would still require less accuracy than traditional highlighting). Once there, they could initiate selection without clicking, which could address mouse slip while clicking, a common problem with advanced age or motor impairment [110, 111, 343]. If issues with tremor lead to lower accuracy, one approach that might be investigated is smoothing mouse movement using generative models trained on a user's individual behavior (e.g., [360]). More generally, additional research is needed to understand the suitability of wiggling across a variety of user capabilities, contexts, and devices [216, 217].

Our lab study had several limitations. Given the short amount of training and practice time, some participants might not have been able to fully familiarize themselves with the wiggle-based collection and triaging techniques offered by Wigglite. The study tasks and topics might not be the ones that participants typically encounter, and therefore they may not have sufficient motivation or background context as in real life. However, we attempted to mitigate these risks by carefully preparing the study setup: (1) we chose the training and real study tasks based on actual product comparison topics that people are faced with; (2) we had participants practice using Wigglite as well as its baseline version for each condition simulating what they needed to do for the real tasks, and (3) we provided participants with ample amount of background information to help them get prepared. In addition, we chose comparison shopping as the topic of both tasks primarily because it is a domain that a large proportion of people (including all of our participants) are familiar with and often perform information collection in. Through literature review and empirical insights from a series of pilot studies, we found that comparison shopping shares many common characteristics with other domains like developers looking for code snippets and patients researching a medical diagnosis, where people spend a significant amount of time online

discovering and making sense of different options, prioritizing which to explore next, and learning about the different trade-offs that make the options more or less suitable for their personal goals. Future work could further address these limitations by having participants use Wigglite for their own work and personal tasks and projects, which would presumably fuel them with the necessary motivation and context and engage with Wigglite in a more organic way.

While sensemaking in various domains might exhibit different characteristics and therefore lead to different information foraging behavior patterns, we chose both the study tasks to be in the domain of comparison shopping to at least make sure that the tasks are roughly of equal difficulty. In addition, product comparison shopping embodies many of the common sensemaking properties and needs that people have, for example, it is information dense so that users would potentially have to read and process lots of information and collect quite a few items, and users would often have to interpret the information based on their own goals and context, so that there is a need for them to externalize their mental context alongside the collected information. Nevertheless, we would like to address this limitation by evaluating Wigglite in a variety of domains where sensemaking usually occurs, such as students conducting literature reviews, patients researching medical diagnoses, and programmers learning unfamiliar APIs.

There was also a risk of participants already being familiar with a topic, such as an expert photographer doing task (A). However, in the post-study interviews, we confirmed that none reported having extensive experience or expertise in any of the task topics.

5.9 Discussion and Future Work

An essential goal of this work is to explore ways to enable people to focus on reading and comprehending actual content rather than splitting attention on the mechanics of collecting information as well as externalizing their mental context. However, prior research [40, 87, 184] has suggested that there is a higher likelihood for people to recall and trust the information if they consciously spend time collecting and synthesizing it with the existing information. This raises an interesting tension and trade-off between pursuing low-cost interactions for information capturing and triaging versus consciously collecting and synthesizing the encountered information — future research would be required to examine the long-term effect of using lightweight systems like Wigglite on people's learning outcomes and decision results in various sensemaking scenarios.

Research on activity-based computing [58, 95] has suggested the benefits of granting users access to their information repository as well as the ability to perform tasks across multiple devices. While the current implementation of Wigglite mobile application does enable users to

collect information and triage it with valence as well as to review their collected information, extending it to support more complex operations such as creating and curating topics could be an interesting direction for future work. We anticipate two challenges: (1) how to reasonably leverage the limited screen real estate to design user interfaces and interactions that feel native to a mobile device while also being functionally lightweight and intuitive enough to perform, and (2) exploring the realistic role of mobile devices in the sensemaking ecosystem [173, 359], i.e., striking a balance between pursuing feature parity with the Wigglite desktop counterpart and designing to specifically support practical use cases (e.g., reviewing and triaging information during a commute).

Though we extended the wiggle gesture to support encoding the valence and priority of information, we envision a larger design space where different aspects and properties of the wiggling movement could be mapped to various functions to increase its expressiveness and utility. For example, the *speed* of each movement, the *duration* of the total movement, or the *size* of the gesture (currently mapped to target size selection) are all continuous measures that may intuitively map to various qualities that could be used in interactions, such as uncertainty or confidence towards some content. Building on what we mentioned in section 5.7.5 and 5.8, future work could investigate: (1) ways to support customizing the mapping of the different aspects of a wiggle according to users' preferences and sensemaking scenarios, and (2) intelligently learning and adapting to users' needs and habits over time, such as re-calibrating the recognition software to account for individuals' cognitive and physical conditions [343].

Last but not least, we envision a future where the wiggling technique as a new class of interaction can be extended and applied to tasks and applications other than sensemaking. On the one hand, a consistent application scenario envisioned by participants involved using wiggling for repeated selection, extraction, and annotation of various types of data as part of a data processing task, such as aggregating recipes online before going shopping, or saving specific torque numbers of fasteners while working on a car. On the other hand, wiggling might also be used for many other system-level behaviors as it does not conflict with most traditional selections or gestures. For example, wiggling with popup or cross-through menus could be applied for quickly modifying device settings on the fly, such as screen zoom or brightness. Additionally, wiggling could serve as an initial activation for a much more expressive set of gestures, or even summon an intelligent agent (such as a voice assistant) that can perform a complex action on the specified item.

Chapter 6

Strata: Evaluating and Reusing Summarized Knowledge

As the amount of information online continues to grow, a correspondingly important opportunity is for individuals to reuse knowledge which has been summarized by others rather than starting from scratch. However, appropriate reuse requires judging the relevance, trustworthiness, and thoroughness of others' knowledge in relation to an individual's goals and context. In this chapter, we explore augmenting judgments of the appropriateness of reusing knowledge in the domain of programming, specifically of reusing artifacts that result from other developers' searching and decision making. Through an analysis of prior research on sensemaking and trust, along with new interviews with developers, we synthesized a framework for reuse judgments (Table 6.1). The interviews also validated that developers express a desire for help with judging whether to reuse an existing decision. From this framework, we developed a set of techniques for capturing the initial decision maker's behavior and visualizing signals calculated based on the behavior, to facilitate subsequent consumers' reuse decisions, instantiated in a prototype system called Strata. Results of a user study suggest that the system significantly improves the accuracy, depth, and speed of reusing decisions. These results have implications for systems involving user-generated content in which other users need to evaluate the relevance and trustworthiness of that content.

This chapter is modified from the following published paper: Michael Xieyang Liu, Aniket Kittur, and Brad A. Myers. 2021. "To Reuse or Not To Reuse? A Framework and System for Evaluating Summarized Knowledge." *Proc. ACM Hum.-Comput. Interact. 5, CSCW1, Article 166 (April 2021), 35 pages.*

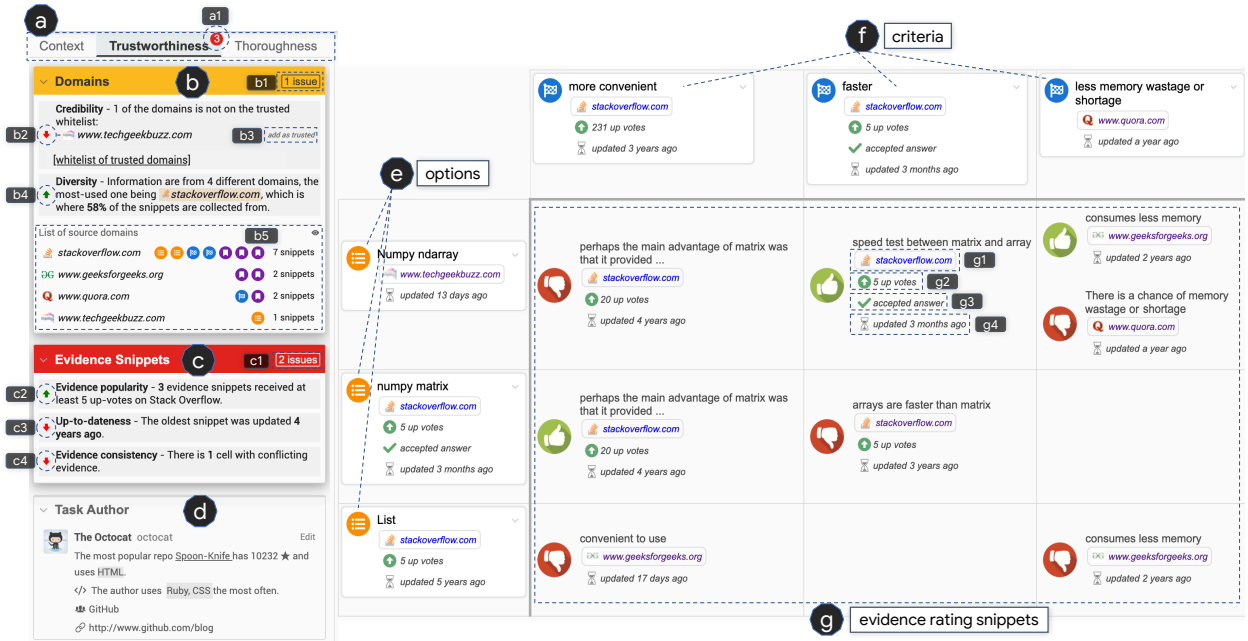


Figure 6.1: Strata's user interface. Strata helps developers evaluate three main facets of appropriateness of reusing a Unakite comparison table with options (e), criteria (f), and evidence (g) through three overview panels: (a) the *Context* panel, the *Trustworthiness* panel, and the *Thoroughness* panel. Each panel contains the *groups* (such as (b), (c), (d)) of appropriateness properties to directly address developers' information needs. Developers will also be alerted of any potential issues with respect to each facet (e.g., b2, c3, c4).

6.1 Introduction

Information and knowledge reuse has become a highly consistent paradigm across a wide range of fields and disciplines to advance their respective frontiers, such as reusing previous engineering best practices on future generations of products [42,43], taking advantage of schemas and results from previous sensemaking episodes to create new representations and understandings of the world [114,192,262,283], and plugging in previously written and well-maintained design patterns and code snippets to build novel software features and functionalities [4,18,122,123,200]. Reusing proven information and knowledge promises the benefits of potentially reduced workload and development cycles [42, 200], improved quality and performance [123, 142, 355], and more time for creation and innovation [171, 242, 246, 355].

There have been various commercial and research information gathering and sensemaking systems that help people with creating reusable knowledge by helping with capturing [27,46,220], organizing [73,144,145], disseminating [82,223,290], and understanding [73,178,260,262,283] information. One that is relevant to the context of programming, our Unakite system [232], enables developers to collect and organize information online into comparison tables with options, criteria, and evidence to help with making decisions (see Figure 6.1-e,f,g). Systems like these often support keeping track of information for sharing with others later [54, 145, 232, 262, 283]. For

example, Unakite might present a comparison table authored by an initial developer (who we call the *author*) to help subsequent developers (who we call the *consumers* or *readers*) pick an API to represent matrices in Python (as in Figure 6.1 and Figure 3.1), or to choose the best JavaScript framework to build a website. Unakite is designed to help consumers reuse the decisions and trade-offs identified by the author [127, 166, 209, 232, 308] instead of spending the time to discover them from scratch.

However, a major challenge to such a knowledge artifact actually being suitable for reuse is that the consumers do not know if it is *appropriate* to use it or not [246, 355]. Prior research suggested that when checking if a piece of online information can be reused or not, people primarily focus on verifying its *correctness*, and often use *credibility* as a surrogate for correctness because it is easier to check and is highly correlated with correctness [162]. For example, signals that can be leveraged to judge credibility include whether the information came from credible sources, whether the way it was presented looked credible, and what the author's qualifications and credentials were [117, 253, 317, 353]. In addition, the correctness of information can not always be measured objectively, but rather often depends on the situation [115]; for example, a statement that a sorting algorithm is "fast" may depend on the size of the data it is applied to. Furthermore, the knowledge artifacts shown by previous systems are usually a collection and synthesis of different individual pieces of information from different sources. They often capture the author's *opinion* about whether a decision should be made in one way or another, and there is not likely to be a single correct answer but multiple valid options with trade-offs [232]. Unlike general web pages and their content, such knowledge artifacts require many more types of judgments in addition to credibility for someone to decide whether it is appropriate to reuse them or not, including whether the goal and context of the author matches that of the consumer's [162, 246], how thorough was the author's research [102, 283], etc.

Another challenge identified in sensemaking research is that, in reality, consumers often opt to start from scratch rather than reusing previous users' work because of the high costs associated with 1) systematically identifying all of the potential aspects of the work to verify, and 2) obtaining access to properties that could help with the verification [114, 228, 229, 253]. For example, when checking the thoroughness of an author's research, the list of search queries used, the web pages visited, the pages that the author spent the most time reading, and the potential alternatives that were overlooked can all be valid properties to help with the assessment, but are currently not kept track of (even by systems such as Unakite) and hence are not available to the consumer.

In this work, we explore these challenges in the context of reusing the comparison tables created using the Unakite system, where the consumer developer needs to evaluate the *appropri-*

ateness of reusing the table authored by the initial developer. We perform our research through a *user-centered design* approach. From the vast body of prior work discussing frameworks and measurements for issues of trust and reuse, we extracted properties of importance to developers. We then conducted formative needs-finding interviews with developers about their information needs when evaluating appropriateness. We then synthesized all this information together, resulting in the three key facets of the author's *context*, and the *trustworthiness* and *thoroughness* of the resulting knowledge artifacts, each with a collection of the consumers' specific information needs, which are summarized in a framework in Table 6.1. We then devised various key signals and properties that can be used to address those needs as well as mechanisms to automatically identify, compute or keep track of them as an author collects information, which are summarized in the last column of Table 6.1. Then, we iteratively designed a hierarchical presentation of the information that lets consumers view and explore those signals and properties interactively, by augmenting the original Unakite tables. These were implemented in a prototype system called Strata[1], which consists of a browser plugin for Google Chrome and a web application (see Figure 6.1). Finally, we conducted a user study to evaluate Strata's effectiveness.

The primary contributions described in this work include:

- a formative **study** showing developers' needs for support with reusing previously-generated knowledge,
- a synthesized **framework** (Table 6.1) for augmenting judgments of appropriate reuse including three major facets: context, trustworthiness, and thoroughness,
- a prototype **system** called Strata that automatically records, computes, and visualizes many of the appropriateness signals described in the framework,
- an **evaluation** of the prototype system that offers insights into its usability, usefulness, and effectiveness.

6.2 Related Work

6.2.1 Information and Knowledge Reuse

As formulated by Davenport et al. in 1996 [93] and Markus in 2001 [246], knowledge processes are often categorized by whether they involve *knowledge creation* (e.g., research and development of new products and services, or writing books or articles) or *knowledge reuse* (e.g., reapplying

[1]Strata is named after a series of layers of rock that shows the history of a geographical location. It stands for "**S**idebar **T**owards **R**euse and to **A**ssess **T**rustworthiness and **A**pplicability".

existing components and best practices to solve common problems). While there is much research into the significance and difficulties of knowledge creation and innovation [93, 144, 146, 192, 199, 270], the effective reuse of knowledge has been shown to be a more frequent strategy and concern to individuals and organizations [93, 100, 246, 272, 275, 378, 379].

Many systems have been developed to support the multiple stages of information and knowledge reuse as mapped out by Markus [246]: *capturing and documenting knowledge*, *packaging and distributing knowledge*, and *reusing knowledge*. Among them, some systems support capturing, organizing, and keeping track of information in the first place (e.g., [46, 145, 224, 225, 232, 347]), some aim to deliver and surface existing knowledge directly to a user without the need of complex matching and frequent context switches (e.g., [54, 74, 290]), and others facilitate the digesting and understanding of knowledge (e.g., [229, 232, 332]). However, having a literal understanding of a knowledge artifact does not by itself imply reuse — a major barrier to that knowledge actually being useful is the consumer does not know whether it is *appropriate* to use it or not [246, 355].

Prior research provides insights into various properties that people look for in order to evaluate the appropriateness for reuse, such as source credibility [97, 108, 116, 253, 317, 341], information currency (or up-to-dateness) [26, 53, 253], information popularity [253, 317], goals and purposes (what the author wanted to achieve) [284, 320], etc. However, much research such as the above focuses on specific issues about the general credibility of web content, while knowledge artifacts previously collected and synthesized by an author require many more types of judgments beyond credibility in order for a consumer to decide its appropriateness for reuse. To the best of our knowledge, there remains no systematic models or frameworks for understanding the factors that affect the judgments of the reuse of previously created knowledge artifacts. Such a framework could be helpful for driving research studying and augmenting reuse across a variety of domains and forms. In this thesis, we take a step towards such a framework, starting with knowledge artifacts in the form of comparison tables, which are widely used, and in the domain of programming, where knowledge reuse happens frequently [54, 142, 156, 166, 198, 207, 232, 290, 327, 327]. In the following sections, we discuss three of the most relevant threads of research as they relate to judgments of knowledge reuse.

6.2.2 Evaluating Online Information Credibility

6.2.2.1 Models and Heuristics for Evaluating Online Information Credibility

One of the most researched facets of knowledge reuse is evaluating online information credibility [117, 253, 317, 353] (or "trustworthiness" [341]), which focuses on facets of authenticity,

reliability, and trustworthiness of a given piece of content online, ranging from e-commerce transactions to online discussions and collaborations [194, 332, 340]. Prior work has employed bottom-up approaches like surveys and contextual inquiries and reported various factors that influence credibility assessment, including but not limited to: domain name and URL, presence of date stamp showing information is current, author identification and indication of his or her expertise, citations to scientific data or references, and user ratings and reviews [26, 53, 108, 116, 125, 251, 253, 255, 323, 341, 353].

In addition, models and heuristics for credibility assessment have also been proposed, for example, the *checklist model*, which guides users through a checklist of critical factors during assessment [253], and the *contextual model*, which emphasizes the use of external information to establish credibility [251], such as promoting peer-reviewed resources and seeking corroborating or conflicting evidence. A summary by Metzger et al. [255] suggests that users routinely invoke *cognitive heuristics* to evaluate the credibility of information and sources online, such as the *reputation heuristic* (checking if the source of the information has good reputation and credentials), and the *expectancy violation heuristic* (checking if a website or its content conforms to their original expectations).

However, in reality, it has repeatedly been shown that people are often underprepared and have trouble determining how to evaluate the credibility of online information [32, 251, 254, 313], which is often deemed to be too much work [251, 318], having a high possibility of missing important details [253, 255], and eventually leading to abandonment, mistrust or misuse [228, 229, 253] of the information. This reflects a significant gap between research and reality: while prior work provides insights into the various factors affecting online information credibility and ways people reason about them, people need tool support that systematically helps with credibility assessment and information reuse. We address this gap by providing a prototype system (the Strata system) that (1) automatically extracts appropriateness signals (including those related to credibility) from the original knowledge content when possible; and (2) processes and presents them to the consumer of the knowledge in a hierarchical visualization that directly addresses their information needs during the evaluation of the appropriateness to reuse.

6.2.2.2 Support for Evaluating Collaboratively-built Knowledge Content

Collaborative knowledge building, exemplified by the Wikipedia project [7] and Stack Overflow [6], has become highly popular in many domains, and its mutable nature that virtually *anyone can edit anything* has invited considerable research into helping users evaluate the trustworthiness of its content. For example, the revision histories [332, 350, 376, 377], review processes [351],

and the external references [117, 121] of an article can be modeled and visualized to help improve transparency and the evaluation of its trustworthiness. In addition, an author's past performance, such as their editing history on Wikipedia or previously answered questions on Stack Overflow, can be mined [25, 323] and surfaced [332] to help knowledge consumers determine the author's reputation, expertise, and other accountability metrics. Encouragingly, Kittur et al. [194] showed that surfacing trust-relevant information from Wikipedia articles had a dramatic impact on users' perceived trustworthiness of those articles, holding constant the content itself.

However, despite the overwhelming importance and increasing research effort, being considered trustworthy is often not the sufficient condition for reuse, nor is trustworthiness always the first facet that users evaluate — research has shown that people often have trouble understanding a piece of information when it is taken out of its original context [232, 246] and figuring out if it is indeed relevant to their own situation [51, 311, 318] before they start to think about trustworthiness and credibility. In addition, they also wonder about how much effort has been put into creating a piece of knowledge and does it cover everything that they are interested in [246, 283, 318, 320, 380] before they can give a final verdict on reusing it or not. Therefore, we draw from and build upon these prior works, where we iterated to identify, extract, and surface not only the important elements of trustworthiness but also context and thoroughness to help consumers make a more comprehensive assessment of the appropriateness of reusing knowledge, exemplified by decisions and their rationale in programming.

6.2.3 Sensemaking Handoff

Much research has explored the activity of *sensemaking handoff*, during which one individual must continue the sensemaking work where another has left off. It frequently happens in asynchronous collaborations [114, 283, 284, 380], shift changes [282], etc., during which the current sensemaker (consumer) needs to make sense of and evaluate the appropriateness of reusing the results generated by a previous sensemaker (author) [246, 318]. Various metadata and properties parallel to the main artifacts of sensemaking have been proposed that would help the people with this process, such as the awareness of the previous sensemaking process [102, 283] (e.g., search queries and visited web pages), the level of expertise of the author [246, 320], and the context of the original sensemaking problem [246].

However, it is both time and effort intensive for an author to keep track of their rationale and processes with little immediate payoff, which is also often for the benefit of others rather than themselves [232]. Even in situations where authors have the explicit wish to help, they are often uncertain of what metadata and properties to provide and how those can be instantiated

using concrete signals that would be valuable to the consumers in evaluating the reusability of their sensemaking results [318]. We address these barriers in the context of reusing decisions in programming by iteratively developing a framework that summarizes the major facets that consumers care about during the evaluation of appropriateness to reuse along with the corresponding detailed information signals, and a set of technical approaches that can automatically extract, compute, and visualize them when possible. We integrated these into our Unakite system [232] that helps authors organize and record their decisions for reuse, saving them the burden of coming up with the appropriate signals to keep track of as well as potential extra effort needed to accurately obtain them.

6.2.4 Knowledge Reuse in Programming

The practice of knowledge reuse has been particularly relevant in the software industry [142]. Code reuse, in particular, has become a hugely successful paradigm in the development of new software products and services in both the commercial and open source sector. Developers frequently use well-maintained functional code modules from code-sharing platforms such as GitHub [2] and npm [4], enjoying the benefits of significantly reduced workload, improved productivity, enhanced software performance, stability and security, and more time for innovation [122, 123, 142, 176, 246, 259, 266, 327].

Despite the fact that software code is the most obvious target for reuse [142, 259, 327], knowledge reuse in programming may go well beyond code, as stated by Barns and Bollinger [39]: "The defining characteristic of good reuse is not the reuse of software *per se*, but the reuse of human problem-solving." Indeed, developers on community Q&A websites like Stack Overflow [6] share not only code examples [54, 290] but also decision making strategies, design rationale such as alternative options, criteria or constraints that should be met, and the resulting trade-offs [166, 232]. Furthermore, questions about design rationale are widely cited by developers as some of the hardest to answer [207, 208, 325]. Tools like Unakite [232] can greatly reduce the costs to keep track of and later understand such rationale knowledge, with the hope that such knowledge can ultimately be better reused rather than be obtained from scratch requiring duplicated research effort [142, 230]. In Strata, we further advance this research thread by developing features and affordances enabling developers to evaluate the context, trustworthiness, and thoroughness of previously-made decisions, which is arguably one of the missing links between understanding and reuse.

6.3 Background and Formative Investigations

Although Unakite has been shown through lab studies to help the initial developer in making a programming decision, it displays few of the signals suggested by the research discussed above on trust and sensemaking handoff that could help consumers of the table decide whether it is appropriate for them to reuse it. For example, the initial table creator may or may not have been thorough in their research; may or may not have the same context and environment; or may or may not care about the same goals as the consumer. Although we use Unakite as a specific context, there are many similar examples of developers creating comparison tables in code documentation, blogs, and Stack Overflow [8, 9], which are typically even sparser in terms of signals for reuse appropriateness, with no supporting interactivity or drill-downs possible.

6.3.1 Formative Interviews

To characterize the prevalence and types of issues developers have with knowledge reuse, specifically with reuse of programming decisions, we conducted semi-structured interviews with 15 developers (5 female, 10 male). Participants were recruited through mailing lists, social media postings, and word-of-mouth. To capture a variety of processes, we chose 8 professional developers, 3 doctoral students, and 4 master students. While we do not claim that this sample is representative of all developers, the interviews informed and motivated the development of the subsequent framework (Table 6.1) and the design of the Strata system.

We began by asking participants about their experiences in reusing someone else's decisions when programming and how frequently would that situation occur in their work. We then explored how they manage these situations and their information needs, in particular, what questions do they have when evaluating the appropriateness to reuse and how answers to those questions may affect their final verdicts on reusability. In addition to eliciting facts on their past experiences, we also presented them with a set of decision tables in the running Unakite application (which were directly adapted from real tables online, e.g., [316]) as well as the corresponding background situational context, and asked them to judge if they could reuse these tables in those given situations. We asked them to speak about any questions they had and perform any inquiry they wanted to answer those questions (e.g., checking the sources, searching for evidence online, etc.). Finally, we wrapped up with questions probing their experience with explaining their design rationale to others, and whether and how do they convince others that their decisions are appropriate to be reused.

Interviews were conducted either in person or remotely by the first author and lasted 30 minutes. They were audio-recorded and then transcribed. In addition, screenshots of participants' computers were taken for later analysis when applicable. Then, we went through the transcriptions and coded them via an open coding approach [77], which included multiple iterations of discussions with the research team. Our key findings are presented below.

6.4 Framework

Data from the formative study suggested that developers would benefit from support in evaluating the appropriateness of reusing decisions. For example, there are many indicators that could be beneficial to surface to help users make these judgments, ranging from the expertise of the author to the quantity and legitimacy of the sources used. Although there has been little prior work characterizing the most important factors for decision reuse specifically by developers, as listed above there has been significant work discussing frameworks and measurements relevant to evaluating and reusing knowledge, such as online information credibility judgment [251,253,254,255], asynchronous collaboration [262, 284], and sensemaking handoff [114, 318, 319, 320]. From these research papers, we extracted properties and signals that would be important and relevant to decision reuse for developers.

By coding and synthesizing the aforementioned prior work as well as the formative study results through affinity diagramming, we identified three major clusters, that we call *facets*, when evaluating the appropriateness for reuse in programming: the original author's decision making *context*, and the *trustworthiness* and *thoroughness* of the resulting decision. We used these as a guide in developing an integrated framework, shown in Table 6.1, consisting of the three identified facets (column 1), specific information needs of developers with regard to each facet (column 2), selected evidence for the importance of these information needs as well as possible solutions to address them from prior work (column 3), and sample quotes from our formative interviews (column 4). These insights together inspired the features for our subsequent Strata system (column 5). We now discuss the framework in detail, along with the support from the prior work and the formative interviews. The design of Strata follows in section 6.5.

6.4.1 Context

Although in prior work the importance of understanding the trustworthiness of information often outshines everything else when evaluating the appropriateness to reuse [194, 246], we were surprised to find out that, at least in the domain of programming decision reuse, developers of-

Facet	Information Need	Selected References in Prior Research	Sample Quotes in Formative Study	Selected Supporting Features in Strata
Context	Goals of the original decision	• Search queries are useful for encoding task goals & contexts in various settings like asynchronous collaborations [46, 262, 283, 284, 320, 375].	• *"This looks like it's trying to pick a speech recognition API, but what I want is actually text to speech."*	• Keeping track of the author's search queries to reflect his or her task goal.
	Explanation or contextualization of information	• Recontextualization of information helps with understanding [232, 246]. • Clarity and informativeness of website content improves understanding [116, 341].	• *"What does this 'very efficient' mean, is it 'memory' or 'time' efficient?"* • *"Is it [a sorting algorithm] 'fast' only when there're a few hundred data points or also when there are millions of data points?"*	• Keeping track of the surroundings along with the information snippets and presenting them as contextual explanations.
	Situational awareness	• Awareness of common ground facilitates sensemaking handoff [84, 318, 320]. • Users need awareness of each others' actions in order to perform their tasks better [30, 262, 265, 283].	• *"I want to solve it with pure JavaScript, but it seems that most of the answers here are actually written using jQuery?"* • *"I'm using Python 2.7 at the moment, which is fairly old, does this example also use this version?"*	• Detecting information about languages, frameworks, and their versions mentioned in information snippets with a predefined yet easily extensible list of detectors.
Trustworthiness	Source credibility and diversity	• Source credibility affects trustworthiness of information [97, 108, 116, 253, 341]. • Sources similar to what a consumer usually uses are more likely to be deemed credible [255, 317].	• *"If it's from Stack Overflow, I'm usually fine with it. But if it's from some random blog posts written by some random guy, I would think twice."* • *"I wonder if all of these just came from the official documentation or there're also other developer forums."*	• Visualizing the distribution of information snippets across different domains (websites). • Alerting consumers of potential untrusted domains.
	Information up-to-dateness	• Information currency affects its perceived credibility [26, 53, 253].	• *"Is this speed comparison [between React, Angular, and Vue] up-to-date now that Angular 9 was just released?"*	• Extracting and surfacing the last updated time of information snippets.
	Information popularity	• People apply *the endorsement heuristic* to evaluate credibility [253]. • People seek social proof when evaluating credibility [317].	• *"If there're a lot of other devs [who] also think this is a good idea, then I'm much more comfortable to use it."*	• Extracting and surfacing signals showing information popularity, such as the up-vote count of an answer on Stack Overflow.
	Information consistency	• People apply *the consistency heuristic* to evaluate credibility [253]. • People seek more than one source to verify information [251].	• *"It claims PyTorch is much easier to learn than Tensorflow, but I wonder if there're people suggesting otherwise."*	• Alerting consumers if there are conflicting (both positive and negative) ratings in any of the table cells.
	Author credibility	• The author's level of expertise affects information trustworthiness [97, 194, 317]. • Disclosing patterns of past performance helps people evaluate trustworthiness [194, 323, 332].	• *"Does the table author know what he's doing?"* • *"Is the author saying all the nice things about Caffe because he has lots of experience with it or because he's biased?"*	• Surfacing credibility and bias signals from the table author's Github profile, such as their primary programming language, number of stars on their repositories, and affiliation.
Thoroughness	Research process and effort	• External representations handed off should indicate prior investigative process and insights [283, 284, 380], how much work had been done, and how mature the representation was [318, 320].	• *"How much effort was put into making this decision?"* • *"What did the author focus on?"*	• Keeping track of and visualizing the author's activities on an interactive timeline view, including search queries, pages visited, duration of stay on the pages, information snippets collected, etc.
	Alternatives or competitors	• Knowledge and sensemaking results should indicate their coverage and scope [97, 253].	• *"I heard anecdotally that Svelte gives you much better performance than all these big (JavaScript) frameworks [React, Angular, and Vue]. I should take a look at that before I decide."*	• Finding and surfacing commonly searched-for alternatives mentioned in Google autocomplete suggestions.
	Usable artifacts	• Developers need help finding and reusing code examples [54, 273, 290].	• *"Which option was chosen in the end?"* • *"[Are there] any code snippets that I can immediately plug into mine and test?"*	• Extracting and surfacing code examples from information snippets.

Table 6.1: A framework summarizing the three major facets (column 1) when evaluating the appropriateness to reuse knowledge, including people's specific information needs (column 2), selected evidence from prior work (column 3), sample quotes from our formative study interviews (column 4), and features we devised to support the information needs in the subsequent Strata system (column 5).

ten ask questions about the *context* of a previously-made decision before they proceed to assess trustworthiness (9/15). Cited reasons include that one needs to know *"how relevant it is to what I am doing"* (P5) first, and if the context of the original decision does not align very well with the problem at hand, one would often stop the evaluation process and move on to look for new solutions. For example, if a developer is working in Java, solutions that only work in JavaScript may not be worth investigating.

6.4.1.1 Goals of the original decision

When evaluating context, most (12/15) participants asked questions about the goals and purposes of the author of the decision in order to compare those with their own. For example, *"this looks like it's trying to pick a speech recognition API, but what I want is actually text to speech,"* (P14) and *"people say they want to do one thing, but after taking a closer look, they really are doing this other thing, which often makes me a tad frustrated"* (P7). Indeed, prior research suggests that the goals of decisions are often treated as "self-evident" given the results, and therefore are often not kept track of by the authors [207, 208]. On the other hand, goal mismatch does not always prevent developers from further evaluating a decision; instead, it can become a *"learning opportunity"* for them to *"know more about a new technology or design pattern"* (P11).

Furthermore, when asked about their experience of making decisions, participants reported that their goals may very well evolve with their exploration process rather than remaining fixed from the beginning (7/15). For example, *"I started out trying to choose a framework to build a mobile app for both Android and iOS, but later I stumbled upon this progressive web app thing that totally fulfills all of my requirements, so I ended up trying to learn more about that, and sort of abandoned the mobile app route that I was originally planning to take"* (P3). This motivated us to develop features (e.g., keeping track of all of the search queries used) to capture not only an author's original goal but also the evolving nature of that goal, so that later knowledge consumers could have a better grasp of how the author's goal changed throughout a decision making process.

6.4.1.2 Explanation or contextualization of information

One of the frustrations that participants reported having is that they often have trouble understanding the meaning of some of the criteria and evidence used in online decision tables (8/15). For example, *"what does this 'very efficient' mean, is it 'memory' or 'time' efficient?"* (P10). In some other circumstances, they suspect that evidence may not hold true when external constraints or requirements change: *"is it [a sorting algorithm] 'fast' only when there're a few hundred data points or also when there are millions of data points"* (P1). Indeed, prior work suggests that clarity and

informativeness of information have a significant impact on how well it is understood [116, 341], and presenting information along with its original context (recontextualization) is considered a good way to help people understand its meaning and the conditions in which it is correct or accurate [115, 232, 246].

In addition, it was also suggested by participants that it is not always easy to recontextualize information, especially when the context is not available (6/15). Unakite partially addressed this by allowing users to create a snippet out of a large block of information in its original HTML format as well as automatically recording the corresponding source URL for later retracing [232]. In Strata, we build on that by introducing the concept of a *context snapshot*, which, at capture time, automatically keeps track of the *surroundings* of an information snippet in addition to the snippet content itself and its source URL. When consumers are reviewing a snippet, they will be able to benefit from the possible explanations such as code examples and performance metrics contained in the surroundings that would otherwise be missing from the snippet content.

6.4.1.3 Situational awareness

An essential part of context is the situation in which the information will be reused. In programming, this corresponds to the languages, libraries, and platforms being used, which are often referred to as *dependencies*, and participants reported checking if a given decision shares the same language or library usage as to what they have to work with (8/15). For example, P7 asked *"I want to solve it with pure JavaScript, but it seems that most of the answers here are actually written using jQuery."* Furthermore, version mismatch has been a frequent issue for reuse in programming. With the continuous rise of the open source software development model [142] and the increasing number of frameworks, libraries, languages, and patterns [3, 5, 13], version and dependency mismatches and errors can cause troubles from missing features to breaking dependent downstream applications [17]. Indeed, participants reported checking for versions before they commit to adopting a certain solution (6/15). For example, *"I'm using Python 2.7 at the moment, which is fairly old; does this example also use this version, or is it using Python 3.5?"* These inspired us to try to automatically detect the language, library, platform, and version information whenever possible when an author collects information online, and surface this to the consumer to directly address their information needs.

6.4.2 Trustworthiness

As mentioned, information trustworthiness or credibility is often used as a surrogate for verifying information correctness [162], and is one of the most reported and researched facets during the

evaluation of the appropriateness to reuse knowledge across many domains [246, 253]. Our interview data shows that it plays a crucial role in the domain of reusing decisions in programming as well.

6.4.2.1 Source credibility and diversity

As suggested by prior work, source credibility has a significant impact on the trustworthiness of information [97, 108, 116, 253, 341]. Not surprisingly, all participants in our study reported this same belief — they are more inclined towards trusting information from sources that are official (e.g., API documentation websites) or with a very good reputation within the community (e.g., Stack Overflow), and are more likely to reject information from sources that they have little experience with, echoing the *reputation heuristic* and the *expectancy violation heuristic* [255, 317] that people generally use to assess trustworthiness. For example, P12 said: *"if it's from Stack Overflow, I'm usually fine with it. But if it's from some random blog posts written by some random guy, I would probably think twice."*

It is worth noting that in addition to credibility, source diversity also plays a role in trustworthiness, according to 7 of the 15 participants. They thought that the more diverse the sources used are, the more likely that the evidence in the table has been *"peer reviewed"* or *"confirmed by a bunch of other devs"*, and *"seeing essentially the same thing independently said on a couple of different sites and forums"* gives them *"peace of mind"*. We believe that source diversity also works in concert with information popularity and consistency, which we will discuss in detail in the upcoming sections. This motivated us to provide source domain information as a direct signal for each of the information snippets collected as well as a visualization of how all the collected snippets are distributed across the different domains, enabling users to easily assess source credibility and diversity.

6.4.2.2 Information up-to-dateness

There was a consensus among the participants that in order to make a correct decision, the evidence used must be up-to-date (11/15). Indeed, prior work also suggests that information currency is another crucial element contributing to its credibility, with the intuition that the older a piece of information is, the more obsolete it gets, which implies a lower level of trustworthiness [26, 53, 253]. This is especially true in today's software development world, where languages and libraries are constantly being updated and older versions are quickly rendered obsolete by newer versions. For example, P6 was keen to stay on top of the state of the art of the JavaScript frontend framework competition: *"Is this speed comparison [between React, Angular, and Vue] up-*

to-date now that Angular 9 was just released?" However, the above heuristic can be taken with a grain of salt by some participants, citing reasons that software that was updated a long time ago does not necessarily mean that it is obsolete. As P4 put it, *"the last release of Haskell was like 10 years ago, but it's still the latest version, and I still use it all the time in my work."* Nevertheless, we elect to provide users with direct access to at least the last updated timestamp information of each snippet that the author collected in an effort to help consumers assess up-to-dateness faster. In addition, the separate information about versions, as mentioned above, allows users to use whichever property is most relevant.

6.4.2.3 Information popularity

Echoing what has been reported in prior work that people seek social proof when evaluating information credibility [253, 317], participants (8/15) said that the popularity of information also plays an important role in its trustworthiness, with the general rule suggesting that the more people that stand behind a solution, the more trustworthy it is. For example, P9 said: *"if there're a lot of other devs [who] also think this is a better idea, then I'm much more comfortable to use it."* This is similar to the *endorsement heuristic* [255], which suggests that people are inclined to perceive information and sources as credible if others do so too. This inspired us to directly present consumers with popularity signals (such as an answer's up-vote number on Stack Overflow, or the number of claps of an article on Medium.com) from where snippets are collected.

Also included in the endorsement heuristic is that people sometimes follow others' endorsements without much scrutiny of the site content or source itself [255]. However, some of our study participants suggest quite the opposite (7/15) — they often put much more emphasis on source credibility over the popularity of specific information snippets from that source. For example, *"in retrospect, if an answer is taken from Stack Overflow, I don't really care about its up-vote number or if it's the officially accepted one, I'll just trust it and use it"* (P3), or *"I don't really look at how many people clapped over a Medium article, the fact that it's from Medium.com is usually good enough for me"* (P8). Though seemingly inconsistent with prior work, we do not claim that this is typical in the domain of programming — one possible explanation is that websites like Stack Overflow by default rank the most up-voted posts at the very top with the specific intention to present the most popular information to readers.

6.4.2.4 Information consistency

In addition to source credibility, diversity, up-to-dateness, and popularity, a few participants (5/15) suggested that having more corroborating evidence implies that a piece of information is

more trustworthy. For example, P6 said: *"This [deep learning library comparison chart] claims that PyTorch is much easier to learn than Tensorflow, but I wonder if there're people suggesting otherwise? I kind of want to see at least one other expert that has experience with both and also says PyTorch is better."* Prior research has also found that people will apply the *consistency heuristic* to evaluate credibility, validating information by checking different websites to make sure that the information was consistent [251, 255]. Meanwhile, consistency also implies the converse — having contradicting evidence will undermine the trustworthiness of an existing piece of information.

6.4.2.5 Author credibility

Prior work has shown that the author's level of expertise impacts the credibility of information [97,317]. This is especially significant in the domain of programming, where there is a substantial difference between novice and expert developers in their experience and ability to evaluate code and libraries [44]. For example, when shown with a comparison table on the topic of choosing a deep learning framework, P11 asked: *"Does the author know what he's doing? I'd rather take advice from someone who's an expert rather than some random undergrad."* However, participants (4/15) also reported that there is no easy way to tell the level of expertise of a table author or if that expertise matches with the topic of the table in the current Unakite system.

Another factor that impacts the credibility of an author is if he or she is biased, possibly due to his or her affiliation or personal preferences — for instance, P12 asked: *"is the author saying all the nice things about Caffe [a deep learning framework] because he has lots of experience with it or because he's biased?"* However, one participant also acknowledged that sometimes these "biases" may not be as negative as it sounds — it could be an indication that an author is highly experienced with one particular option and therefore gives favorable evidence for it. To address the above concerns, prior research suggests that disclosing patterns of an author's past performance may be a good indication of his or her expertise as well as possible biases [194, 323, 332]. This motivated us to at least allow the author to provide a link to his or her GitHub profile, and Strata will automatically compute and show relevant expertise metrics (contribution activities, most proficient programming languages, etc.) and affiliation information to the consumer.

6.4.3 Thoroughness

Another important facet when evaluating the appropriateness to reuse knowledge is thoroughness, which deals with the process and the amount of effort used when creating the knowledge, its coverage and scope, as well as any usable artifacts discovered or produced in the process.

6.4.3.1 Research process and effort

Prior work in sensemaking handoff recommends that when knowledge is handed-off from the author to the consumer, it should let the consumer be aware of the prior investigative process and insights [283, 284, 380], such as how much work has been done, and how mature the knowledge representation is [318, 320]. We also found relevant evidence from the interviews: three participants recalled similar experiences where they learned that the previous decision makers spent little time on exploring the decision space, and therefore the results were *"too immature to be picked up and reused"* and *"missing obvious criteria that you should definitely not leave out"*, and they ended up choosing to ignore those previous decisions and started from scratch to conduct their own research instead. This motivated us to automatically keep track of some of the authors' actions as they create tables using Unakite, such as the search queries used, the pages visited, the duration of their stay on each page and each query, etc. We then use these data to compute key statistics as well as timelines and visualize them to the consumers to help them better understand the author's research and exploration process.

P9 also envisioned that having a holistic understanding of the author's process would give her the ability to parse out the author's intention and focus (which may shift throughout the process, as discussed earlier), and therefore provide hints about what she needs to focus on next if she were to reuse this table as the basis for her own decision.

6.4.3.2 Alternatives or competitors

In addition to the process and effort, prior research recommends that knowledge and sensemaking results should also make apparent their coverage and scope [97, 253], for example, what alternatives have been considered, since not all options will necessarily appear in a Unakite table (especially when the author thinks one does not fit his or her particular needs and is therefore not worth further investigation). However, this does not necessarily imply that the option is inferior for the consumer. In our study, a few of our participants (6/15) were also interested in knowing what would those alternatives (or competitors) be and how they compare with the existing options before they could know if it is appropriate to reuse a table. For example, *"I heard anecdotally that Svelte gives you much better performance than all these big (JavaScript) frameworks [React, Angular, and Vue]. I should take a look at that before I decide. Or maybe there's again something else?"* (P14). This motivated us to take advantage of the Google Autocomplete API to automatically obtain commonly searched-for alternatives to the options that are already in the table, and present these alternatives to the consumers.

6.4.3.3 Usable artifacts

Lastly, participants (10/15) stressed the need for code examples and other usable artifacts from a decision, just as prior work reported that developers need help finding and reusing code examples [54,55,273,290]. For example, P2 directly asked for code examples and the author's chosen option when presented with a decision table on various Java AST parsers: *"[are there] any code snippets that I can immediately plug into mine and test? Or if you can tell me which is the one that the author used, I'll just try that one first."* A few (3/15) participants also suggested that quickly trying out code examples to see if they work or not supersedes almost all other information needs. However, we do not claim this is typical, and later follow-up exchanges with these participants revealed that a vast majority of their current work is low-level detailed implementation, where making sure the code works is of paramount importance. Nevertheless, we implemented techniques to automatically extract code blocks from various snippets and present them to consumers. In addition, we also detect authors' copy events in the browser, and use those as the basis for a heuristic to tell which option the author chose for the decision.

6.4.4 Summary

We found that when evaluating the appropriateness to reuse a piece of knowledge, one should not only assess its trustworthiness (as the majority of the prior research has focused on), but also check for its context and thoroughness. However, no previous system has made significant attempts to address developers' specific information needs with regard to all three of these facets, or to extract appropriateness properties from the original content and present them to the consumer of the knowledge to facilitate reuse. In addition, this process should not put much burden on either the author or the consumer [232, 344] by requiring them to manually locate those appropriateness properties, suggesting the need for largely automatic mechanisms.

6.5 Strata Design and Implementation

Based on the findings in our interviews and the framework, we built a prototype system called Strata to visualize properties and signals of the appropriateness to reuse for the consumers of a decision.

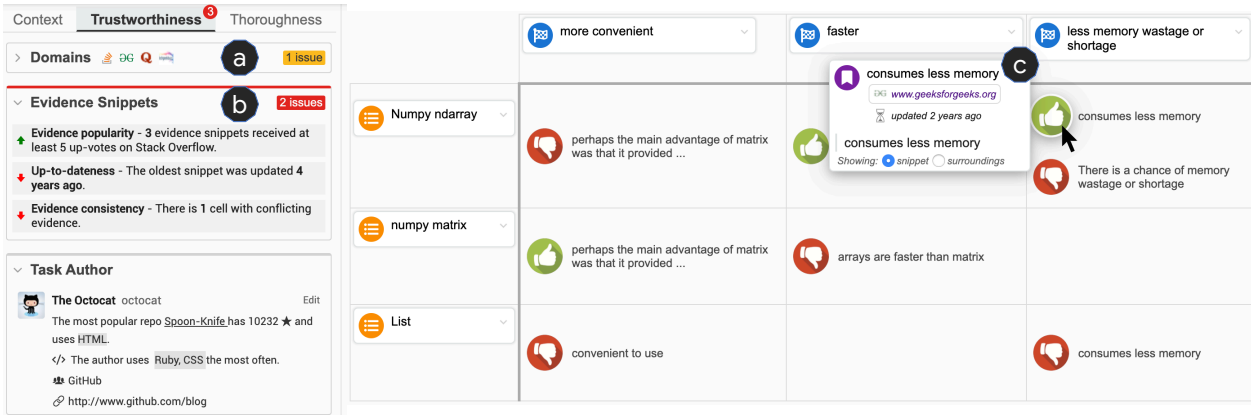


Figure 6.2: On Strata startup, none of the groups are activated to keep the Unakite table on the right clean and concise. Groups can also be collapsed to keep the sidebar interface clean (such as (a)). Mousing over each snippet in the table will only show the exact content that an author captured by default (c), the same as the original Unakite system, rather than the automatically captured *context snapshots*. Only after a user activates some groups in the Strata sidebar (by clicking on their titles) will the corresponding additional metadata appear on the snippets in the table, as shown in Figure 6.1.

6.5.1 Core Design Process and Rationale

We first consulted the interview data and brainstormed the various signals and properties that would theoretically address each of the information need listed in Table 6.1, column 2. Some information needs can be directly addressed by obvious signals, such as surfacing the domain names of the source web pages to consumers so that they know where the information in the table were collected from and if those sources are credible. For information needs that would require explicit effort from the table author to provide, such as the goal of a decision, we also consulted prior literature as well as brainstormed about potential indirect signals that can be used by consumers to infer those needs. For example, search queries are useful for inferring task goals and contexts of an author [46, 262, 283, 320].

In order to obtain these signals, we then built tracking techniques to automatically keep track of the author's activities in the browser while searching and browsing during the creation of a Unakite table. Many of these tracking and extraction techniques use heuristics that are based on the current design of websites that developers most often use, such as extracting the number of up-votes for an answer on a Stack Overflow page. These are meant as a proof-of-concept, and more elaborate and crowd-sourced extraction techniques could be added in the future.

We then set out to design a visualization that presents the consumers with these signals and properties. During our exploration of the design space, we struggled with a fundamental tension between consumers' awareness of all the signals and consumers' limited attention bandwidth. In our initial prototypes, we placed all the signals (approximately 15) in a scrollable vertical list to the

left of the original Unakite table. Users would also be able to hide a signal if it was not relevant. We hoped to make the users aware of all the signals that Strata can provide and give them complete freedom to explore them as they wish. Another rationale for this design was that users would be able to use a combination of signals to fulfill a single information need, for example, both the search queries and the pages visited will help indicate the author's research process and effort, as evidenced by the formative interviews. However, by implementing and testing these design probes with a convenience sample of 8 developers, we realized that having *"everything all at once"* can be overwhelming to the consumers, and they would prefer to just examine one facet at a time and tune out the "noise" (signals that are irrelevant to the facet currently being examined). In addition, we found that there was a disconnect between the signals we showed in the list on the left and the actual content in the table on the right, causing consumers the additional mental burden of trying to match them up. Showing the signals in context along with the various information snippets in the table seemed to be a much better design to address this problem.

These findings guided us towards a hierarchical visualization design of Strata's consumer-facing user interface: to structure these properties and guide the consumers through their evaluation process, we designed Strata as a sidebar to a Unakite table. Strata's sidebar contains three tabbed overview panels for the three facets in the aforementioned framework (Figure 6.1-a). Each overview panel provides multiple *groups* (e.g., Figure 6.1-b,c,d) of appropriateness properties to directly address consumers' information needs as summarized in the framework. In addition, by activating one or more of the groups (by clicking on their titles in the sidebar), consumers will be able to view additional information specific to each snippet in the table. For example, Figure 6.2 shows a state where none of the groups are activated. After activating the Domains group and Evidence Snippets group, consumers will be able to see for each snippet: where it originated (Figure 6.1-g1), how popular it is (Figure 6.1-g2,3), and how old it is (Figure 6.1-g4). This is designed to provide consumers with a high-level overview of each of the facets of reuse as well as the ability to dive into the parts of interest, as recommended by Shneiderman [322]. It is also inspired by the *lens* interaction [48, 74] where the same table content is addressed from three different perspectives.

Like Unakite, Strata consists of an extension to the Chrome Web browser and a web application. Strata's Chrome extension implements the aforementioned new tracking techniques on top of the Unakite Chrome extension. The Strata web application is implemented in HTML, JavaScript, and CSS, using the React JavaScript library [109] as the primary frontend UI develop-

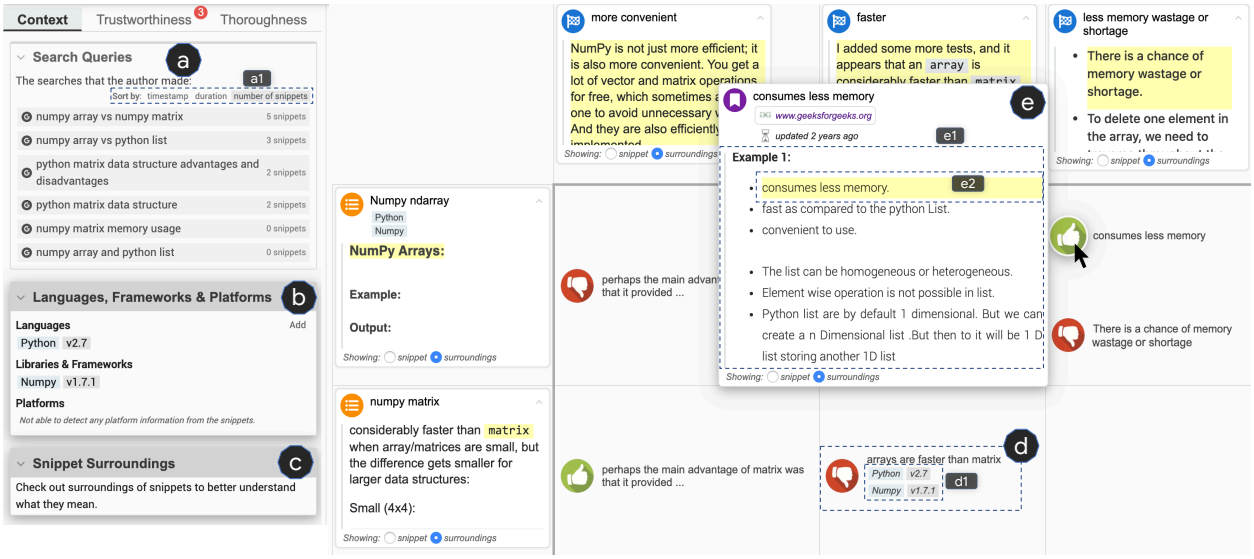


Figure 6.3: Strata's *Context* panel. Consumers are able to check the search queries (a) that the author used to understand his or her goal, examine the languages, frameworks, platforms, and their versions of the snippets (b, d1), and view the surroundings of a snippet through the automatically captured context snapshots (e1).

ment framework and Google's Firebase on the Google Cloud for data management and synchronization as well as user authentication.

We now discuss how the different features in Strata support the three facets listed in the previous framework, and how they are implemented.

6.5.2 Context

6.5.2.1 Capturing goals with search queries

First of all, Strata automatically keeps track of authors' search queries used in Unakite tasks as well as the duration of time they spent on each and the number of information snippets they collected. The duration information is approximated by comparing the timestamp when the next query is issued to that of the current one. It also automatically leaves out any idle time (i.e., time where there is no activities detected in the browser, by monitoring mouse movements, keyboard input, etc.) that are longer than a certain threshold to make the duration approximation more accurate. The idle threshold was empirically tuned to be 8 seconds based on data obtained through pilot testing, and can be flexibly adjusted in the future. For consumers, Strata visualizes these search queries as a list (Figure 6.3-a) to help consumers understand the goals of the task author. They can use the sorting mechanisms at the top (Figure 6.3-a1) to sort the search queries by chronological order, by duration, or by the number of information snippets yielded from each (which is the default sorting order, where ties are broken by ascending chronological order).

There are several advantages of using search queries as a representation of an author's goals. First, they are direct translations of what an author thinks and intends to do to satisfy their information need [309] — for example, issuing the query "numpy matrix vs list" implies that the author would like to find out the differences between the two options. Second, unlike the original Unakite where an author sets the single task goal (as the name of a task) at the beginning, keeping track of all of the search queries (in temporal order) captures not only the author's original goal (which usually is the first query based on pilot study data) but also the evolving nature of the goal (as identified in the formative interviews). Third, the number of snippets yielded from each query serves as an approximation of an author's effort spent on that particular part of the task, which informs consumers of the author's focus throughout the decision making process.

6.5.2.2 Contextualizing information with automatic context snapshots

To help consumers contextualize and understand the meanings of options, criteria, and evidence in Unakite (identified as one of participants' frustrations), Strata introduces the idea of automatically keeping a snapshot of the surroundings of a piece of content called *context snapshot* (inspired by [167]) as an author collects information snippets. Strata uses Unakite's *snapshot* feature, where website content can be captured and preserved with its original styling, including the rich, interactive multimedia objects supported by HTML. The bounds of the surroundings are by default defined as the main content (Strata automatically tries to exclude any advertisements and other forms of injected content on a website) in the visible area of a web page in the browser window. In addition, due to the popularity and importance of Stack Overflow in the domain of programming, we specifically optimized this feature to include not only the particular answer block an author collects information from but also the original question block regardless of whether they are within the bounds, which provides consumers with extra context information. Similar optimizations for other popular developer sites, such as the official documentation, could be added in the future. On the consumer side, by clicking on the title of the *Snippet Surroundings* group (Figure 6.3-c) in the Strata sidebar, consumers will be able to view and scroll through the surroundings for each snippet (Figure 6.3-e1), with the content that the author specifically collected highlighted in yellow (Figure 6.3-e2).

This feature offers several benefits to both the authors and the consumers. The surrounding of a snippet is highly likely to include explicit explanations (such as screenshots, code examples, and execution results) that can help consumers understand exactly what a snippet means. For example, the *Python Lists VS Numpy Arrays* article [19] where a criterion snippet "more efficient" was scooped from, also gives examples of how the two data structures allocate memory blocks

under the hood, suggesting that the author actually meant "more **memory** efficient" rather than "more **time** efficient". Unlike in Unakite, where an author needs to specifically include that entire paragraph when creating a snippet and then manually change the title of the snippet into "more memory efficient" (which may disrupt the workflow), Strata will automatically capture that helpful paragraph into the snippet's context snapshot. During the evaluation of context, consumers will be able to directly view a snippet in its surroundings through its context snapshot without frequent switches to the corresponding original web page to find where the content where the snippet was taken from (which is exactly what participants reported doing in the formative study).

6.5.2.3 Detecting languages, frameworks, and their versions

Strata tries to automatically detect the languages, frameworks, platforms, and their versions used in the snippets to directly address consumers' information needs. To ground this feature, we picked the top 10 of each of the most popular languages, frameworks, and platforms from the 2020 Stack Overflow developer survey [20] and built *detectors* for them. The detectors for a language (or a framework, platform, etc.) is implemented as a set of manually devised keywords (e.g., language statements, special variables, file extensions, etc.) that can uniquely identify the usage or presence of that language. For example, "`es7`", "`console.log`", "`setTimeout`", etc. can be used to identify *JavaScript*, and "`useState`", "`componentDidMount`", "`findDOMNode`", etc. and be used to identify the *React* library. Keywords that can cause ambiguities are specifically avoided, such as "`$`" (the dollar sign) is simultaneously a way to refer to variables in *PHP* and a shortcut for *jQuery*. Strata then automatically tries to find these detectors through optimized string matching in a snippet upon its collection. If there is no hit within the snippet content, Strata will make a second attempt with the content of the snippet's parent web page. Subsequently, Strata uses regular expressions to find version numbers in the vicinity of detected languages, frameworks, and platforms (e.g., "`Angular 9`", "`Python 3.5`", "`React 16.13.1`", etc.) or in the web page's URL (e.g., Java SDK version numbers are encoded in the URL of its official documentation website). In an informal evaluation using materials containing only the currently supported languages, this mechanism was able to successfully extract language information 100% of the time and correctly identify the version information 96% of the time. In the future, one might imagine Strata pulling detectors from open-source detector repositories built, verified, and maintained by the community, which can improve their quality, precision, and recall, or at the very least, letting authors add or correct wrongly detected versions. On the consumer side, this detected information is then presented directly on the corresponding snippet cards in the table (Figure 6.3-d) as well as aggregated in the *Languages, Frameworks, and Platforms* group (Figure 6.3-b).

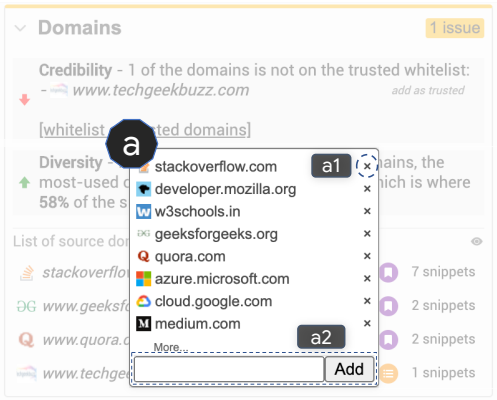


Figure 6.4: The *trusted domains whitelist*. Consumers can remove (a1) or add (a2) a certain domain from the list.

Directly surfacing these version entries to consumers will help them quickly understand the technologies used in the task as well as the specific versions each snippet uses at a glance, to support comparing those with their own situation. For example, one developer would be easily able to figure out that the example code collected by the other developer uses Python 2.7 and therefore does not match with his or her own environment, which uses Python 3.5.

6.5.3 Trustworthiness

To help consumers evaluate the trustworthiness of a table, Strata provides visualizations of various properties that directly address their information needs listed in the framework (e.g., source credibility, information popularity, etc.). Prior work has suggested that surfacing issues or problems that could cause distrust is an effective way to alert and guide users' attention during credibility evaluations [253]. Therefore, in addition to visualizing the trustworthiness properties, we remind users of potential issues that could negatively impact a table's trustworthiness by marking them with a red downward arrow (Figure 6.1-b2,c3,c4). The count of the number of issues is shown in a colored badge on the top-right corner of the Trustworthiness panel (Figure 6.1-a1), with one issue having a yellow color, and more than one issue having a red color (these user-adjustable levels were empirically determined). Future development might explore more sophisticated weighting of the issues beyond counting them equally.

6.5.3.1 Visualizing source credibility and diversity

As shown in Figure 6.1-b, Strata visualizes the distribution of the snippets across different domains (websites) (Figure 6.1-b5), giving consumers a high-level overview of the provenance of

the information in the table. In addition, each snippet in the table is also marked with its domain (Figure 6.1-g1), giving consumers a detailed understanding of where each snippet originated.

Strata also alerts consumers of potential untrusted domains by checking the presence of each domain on a user-defined *trusted domains whitelist*, and flags the ones that are not on the list. For example, a consumer will be able to immediately notice that one of the websites that the author used to collect evidence, `techgeekbuzz.com`, is not on his or her own trusted domains whitelist (Figure 6.1-b2). Currently, the default whitelist was generated by mining and aggregating the websites that 5 full-stack developers (who work for different technology companies and routinely use a variety of languages and technology stacks) visited from their browsing history. We then had them each annotate the websites as either "credible" or "not credible", and removed the ones that they did not all agree upon. This resulted in 25 domains that are considered "credible", including community Q&A sites like `stackoverflow.com`, official documentation sites like `angular.io`, and blog sites like `medium.com`. Domains that sometimes contain non-objective and low-quality information are rejected, such as `reddit.com`. We by no means claim this is complete nor that it applies to everybody — instead, it serves as a starting point and the consumers are able to add and remove items themselves (Figure 6.4-a1,a2). They can also use the "add as trusted" button (Figure 6.1-b3) to add a flagged website to the whitelist so that any future information originating from that website will not be considered as an issue. In the future, one can imagine taking advantage of a larger consumer base and automatically marking websites as trusted if a majority of the consumers have it on their whitelist. We also expect to periodically update the default whitelist over time, as new programming technologies are created and become popular in the future.

To help with the evaluation of source *diversity*, Strata also alerts consumers when there is only limited sources used to construct a table. Currently, Strata considers that there is an issue in terms of source diversity if all of the information comes from one single source (reported by participants in the formative studies as the worst scenario). If that is the case, the green upward arrow for source diversity in Figure 6.1-b4 will become a red downward arrow, reminding consumers that it is an issue. However, this threshold can be set by individual consumers, which would then apply to all future table evaluations they perform. Similar to source credibility issues, this can also be resolved or dismissed by individual consumers if they do not think it is problematic.

6.5.3.2 Examining evidence trustworthiness

Consumers will be able to get information about the popularity, up-to-dateness, and the consistencies of the evidence by activating the *Evidence Snippets* group (Figure 6.1-c).

Each snippet in the table will be marked with signals showing its popularity depending on the websites and pages that it originates from. For example, if a snippet is collected from a Stack Overflow answer post, Strata will automatically extract and show the up-vote number of that post (Figure 6.1-g2) as well as if that answer is the officially accepted answer (Figure 6.1-g3). If a snippet is collected from a Medium.com article, Strata will show the number of claps that article had at the time of collection. We designed this feature to closely fit developers' current ways of evaluating popularity, as reported in the formative studies. Strata will also display an alert in the Evidence Snippets group if some of the snippets in the table have particularly low popularity, such as down-votes on Stack Overflow. As with the other kinds of detectors, we envision these being augmented over time based on where developers are mostly getting their information from.

Unlike the original Unakite, which only showed *when* information was collected (reported as *"not exactly helpful"* by participants in the formative interviews), each snippet in the table will be marked by Strata with the timestamp of when its parent webpage (or answer post if it is from Stack Overflow) was last updated (Figure 6.1-g4). Strata uses a combination of techniques to extract the last updated timestamp information, including using regular expressions to look for date strings in website source code and taking advantage of the JavaScript `document.lastModified` variable (only when the website is static). This serves as a direct measurement of the age of information, and gives consumers an idea of how old the information is. Our study participants also mentioned that they often had trouble quickly locating when articles or blogs are updated online as these timestamps are often displayed in less salient font styles or not visible at all. In addition, Strata will flag snippets that are older than 3 years as a potential issue in the Evidence Snippets group (Figure 6.1-c3), which, similar to other issues, can be manually adjusted or dismissed by the consumer.

Finally, Strata provides initial support for information consistency by informing consumers if there are corroborating or conflicting evidence snippets in a table cell (e.g., there are simultaneously both thumbs-up and thumbs-down ratings for "`numpy ndarray`" causing "less memory wastage or shortage") (Figure 6.1-c4). The culprit table cells with conflicting evidence will be highlighted by mousing over the issue in the Evidence Snippets group, addressing concerns from participants in the formative studies about how such contradictions could be overlooked once a table gets larger with more evidence ratings.

6.5.3.3 Surfacing properties about author credibility

Strata provides consumers with help in evaluating author credibility by allowing authors to manually provide information about themselves. In the current implementation, a table author can input a link to their GitHub profile, and Strata will automatically present the author's name,

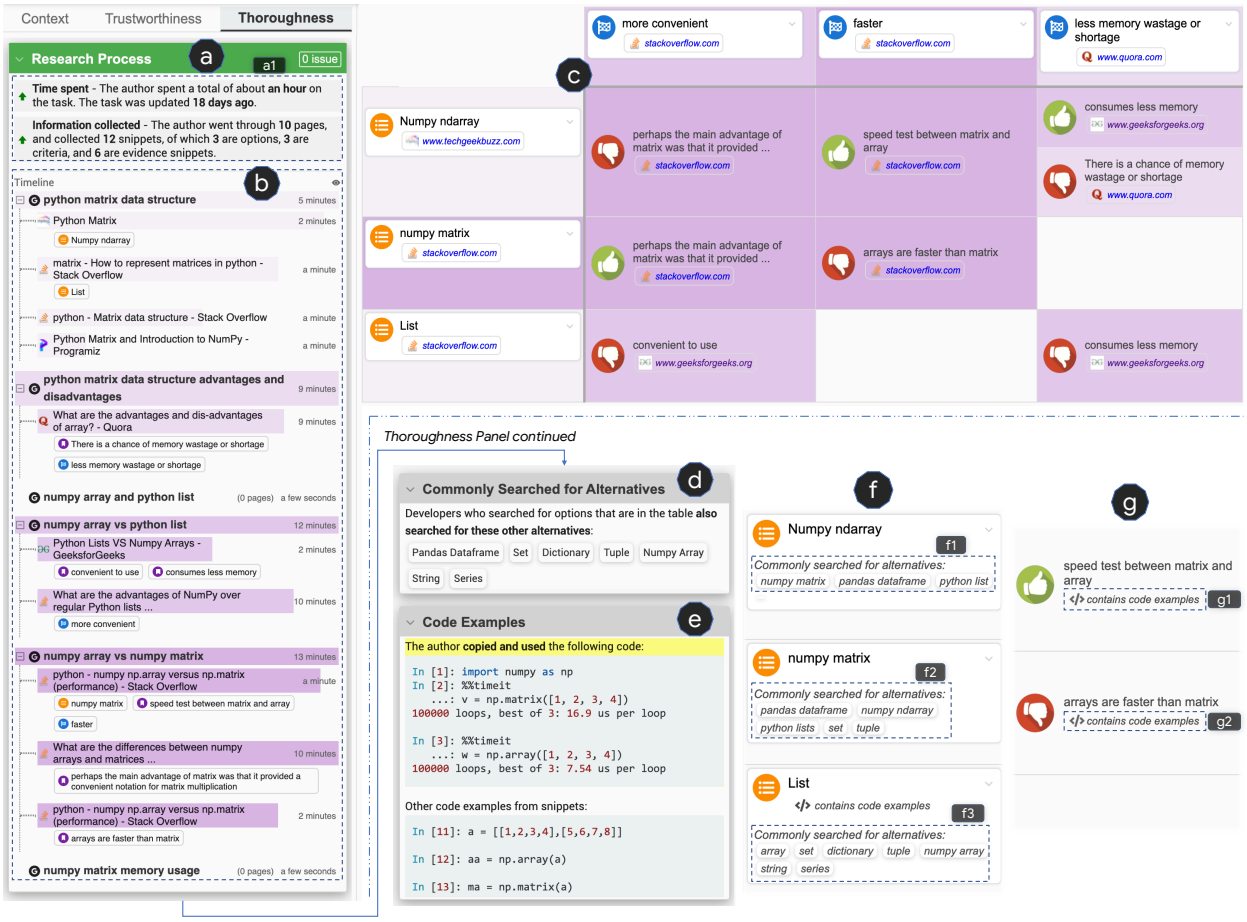


Figure 6.5: Strata's *Thoroughness* panel. Consumers are able to understand the author's research process (a) with the help of the timeline view (b) (a lighter violet means older chronologically), check commonly searched for alternatives to the existing options (d, f1, f2, f3), and check the code examples in the snippets (e).

numbers of stars on the most popular code repositories he or she owns, most used programming languages, affiliation, and a link to his or her GitHub profile page in the *Task Author* group (Figure 6.1-d). We opted to let authors voluntarily provide this information in order to give them the option to protect their privacy and identity. In the future, we will work on mechanisms to automatically perform author modeling in a privacy-preserving way — one idea is to analyze the topics of Stack Overflow questions and coding forums that an author frequently visits to infer his or her expertise. We will also provide an option for authors to provide certain information to consumers anonymously.

6.5.4 Thoroughness

6.5.4.1 Understanding the research process

In order to provide consumers with a clear understanding of an author's research and exploration process, Strata automatically keeps track of several of the author's activities in the background — in addition to the search query tracking discussed earlier, Strata also automatically records the web pages visited, as well as the time spent, progress made (approximated by tracking the percentage of a page that has been scrolled into the visible viewport using JavaScript's `window.onscroll` event), and the number of information snippets collected on each of the web pages.

With these activity data, Strata computes the duration of time the author spent working on a task, the length of time since the task was last updated by the author, and the numbers of options, criteria, and evidence snippets that the author collected (Figure 6.5-a1).

In addition, Strata visualizes the activity information on a timeline view (Figure 6.5-b), which provides an integrated chronological representation of the author's entire research and exploration process during a task. The timeline view is organized with two levels of hierarchies: first by the search queries, and then by the pages that are visited during a particular search. The timeline view is color-coded by different shades of a violet color, with increasing intensity indicating the chronological order (a lighter violet means older). The same color scheme is also applied to the background of the table cells (Figure 6.5-c) when the *Research Process* group is activated. The timeline view is also interactive, mousing over a search query or a page will highlight its corresponding information snippets in the table, together with the colored background, giving consumers an understanding of how the table was constructed chronologically.

6.5.4.2 Suggesting alternatives

Another way for Strata to help with the thoroughness evaluation is to provide consumers with *commonly searched for alternatives* to each option (Figure 6.5-f1,f2,f3). For every option in the table, Strata will automatically obtain the potential alternatives to that option by making Google search queries in the form of "[option_name] vs" or "[option_name] versus" and obtaining a list of top 10 auto-complete candidates using the Google Autocomplete API. This will then be transformed into the *alternatives list* for the corresponding option by extracting and cleaning the part after "vs" or "versus" for each auto-complete candidate, followed by aggregating and removing duplicates. The results are presented in the *Commonly Searched for Alternatives* group (Figure 6.5-d). These alternative lists are generated on the spot every time a table is being reviewed, making sure that Strata always presents the latest information.

This approach offers several benefits to the consumers of the table. First, it offers insights into the popularity of the existing options in the table — if an option (such as "React") appears in all other options' alternatives lists (such as for "Angular" and "Vue"), it suggests that this option has a high popularity. Second, it provides consumers with an understanding of the coverage of the author's research process as well as guidance on potential new opportunities to explore next — if an item (such as "pandas dataframe" in Figure 6.5-d) frequently appears in the existing options' alternatives lists (and therefore will rank higher in the aggregated list in the Commonly Searched for Alternatives group), it suggests that this item might have been overlooked by the author initially, or it might not have been available back when the table was made, and the consumers can focus their investigative effort on it next before deciding whether to reuse this table. This feature could help authors as well, offering real-time reminders of the coverage of their research process and possible new options to consider as they are making decisions.

6.5.4.3 Presenting usable artifacts

Finally, Strata automatically detects and extracts any code examples included in the collected snippets and presents them in the *Code Examples* group under the Thoroughness panel (Figure 6.5-e). This provides consumers the opportunity to directly examine and try out any code examples involved first without diving deeper into the table. In addition, when the Code Examples group is activated, a *"contains code examples"* badge (Figure 6.5-g1,g2) will appear on snippets that contain code examples, helping consumers quickly locate potential code examples for a particular option or criterion in the table.

6.6 Evaluation

We conducted a lab study to evaluate the effectiveness of the framework and the prototype Strata system in helping developers evaluate the appropriateness of reusing decisions.

6.6.1 Experiment Design

6.6.1.1 Participants

We recruited 20 participants (13 male, 7 female) aged 22-37 ($\mu = 26.95$, $\sigma = 3.81$) years old through emails and social media. The participants were required to be 18 or older, fluent in English, and experienced in programming. Participants on average had 8.3 ($\sigma = 3.3$) years of

programming experience, with 11 of them currently working or having worked as a professional developer and the rest having programming experience in universities.

6.6.1.2 Procedure

Participants were presented with 3 tasks in random order. The topics of the tasks were: *(a) choosing a python data structure to represent matrix-like data* (referred to as *Python* from here on), *(b) choosing a deep learning framework to build neural networks* (referred to as *Deep* from here on), and *(c) choosing a cloud computing service to build a video-streaming application* (referred to as *Cloud* from here on). For each task, participants were told what to pretend their background and context was, and they needed to read a table and answer questions about: (1) how much do they think the table is relevant to their given background and context; (2) how much do they trust the content of the table; and (3) to what extent do they think the research effort put into making the table is thorough. Participants were required to list out specific reasons to justify their evaluations.

The study was a between-subjects design, where participants were randomly assigned to either the Strata condition or the Unakite (control) condition. In the Strata condition, participants had full access to all the Strata features described above (along with the table produced by Unakite), while in the Unakite condition, these new features were turned off, so the participants saw only the table, and snippets in the table only showed their titles, contents, timestamps of collection, and links to their original web pages. We imposed a 10-minute limit per task to keep participants from getting caught up in one of the tasks. However, participants were instructed to inform the researcher when they thought they had finished the task or felt like they could make no further progress.

We chose Unakite as the control condition as opposed to raw (and textual) comparison tables online to make sure both conditions had a similar user interface to work with. It also makes the comparison between conditions more realistic — since the original Unakite is already keeping track of where snippets are collected, participants in the Unakite condition would have the ability to go back to the source to examine the appropriateness signals (such as up-vote numbers, last-updated timestamp, etc.) if they wanted to.

Each study session started by obtaining the proper consent and having the participant fill out a demographic survey. Participants in the Unakite condition were given a 10-minute tutorial showcasing the various features of the Unakite web application as well as a practice task on the topic of "choosing a JavaScript frontend framework" before starting. Those in the Strata condition were given a same-length tutorial as well as the same practice task but in Strata instead. At the

	Time	n_{Total}	n_{Valid} for Context	n_{Valid} for Trustworthiness	n_{Valid} for Thoroughness	n_{Valid}	$n_{\text{High Quality}}$	Precision	Recall
Unakite	484.2 (37.8)*	5.20 (0.92)*	1.50 (0.53)	1.30 (0.48)*	1.20 (0.42)*	4.00 (0.67)*	2.90 (0.57)*	55.7% (4.9%)*	24.2% (4.7%)*
Strata	328.2 (48.1)*	7.90 (1.91)*	1.50 (0.53)	3.20 (0.79)*	2.70 (0.82)*	7.40 (1.51)*	7.10 (1.45)*	90.1% (6.8%)*	59.2% (12.1%)*

(a) Python ($n_{\text{Ref. High Quality}} = 12$)

	Time	n_{Total}	n_{Valid} for Context	n_{Valid} for Trustworthiness	n_{Valid} for Thoroughness	n_{Valid}	$n_{\text{High Quality}}$	Precision	Recall
Unakite	393.4 (50.9)*	5.70 (1.06)*	1.70 (0.48)	1.60 (0.70)*	1.40 (0.52)*	4.70 (0.82)*	3.20 (0.92)*	56.1% (12.4%)*	29.1% (8.3%)*
Strata	276.2 (68.3)*	7.80 (1.87)*	1.70 (0.67)	3.00 (1.15)*	2.60 (0.70)*	7.30 (1.83)*	6.90 (1.97)*	88.1% (9.7%)*	64.5% (17.4%)*

(b) Deep ($n_{\text{Ref. High Quality}} = 11$)

	Time	n_{Total}	n_{Valid} for Context	n_{Valid} for Trustworthiness	n_{Valid} for Thoroughness	n_{Valid}	$n_{\text{High Quality}}$	Precision	Recall
Unakite	420.4 (58.9)*	6.20 (1.03)*	1.40 (0.51)*	1.90 (0.74)*	1.50 (0.53)*	4.80 (1.14)*	3.60 (0.97)*	58.5% (15.2%)*	30.0% (8.1%)*
Strata	271.8 (35.3)*	9.60 (2.37)*	2.60 (0.84)*	3.80 (0.92)*	2.60 (0.70)*	9.00 (2.00)*	7.90 (1.45)*	83.8% (8.5%)*	65.8% (12.1%)*

(c) Cloud ($n_{\text{Ref. High Quality}} = 12$)

Table 6.2: Lab study results. The numbers of gold standard high quality reasons for each task, $n_{\text{Ref. High Quality}}$, are listed in their respective captions. We report the mean and standard deviation for: (1) the **time** in seconds taken to finish a task; (2) the total number of reasons participants came up with, n_{Total}; (3) the number of valid reasons, n_{Valid}; (4) the number of high quality reasons, $n_{\text{High Quality}}$; (5) the precision of high quality reasons, calculated as $n_{\text{High Quality}}/n_{\text{Total}}$; (6) as well as the recall of high quality reasons, calculated as $n_{\text{High Quality}}/n_{\text{Ref. High Quality}}$. Statistically significant differences ($p < 0.05$) through t-tests are marked with an *.

end of the study, the participant was invited to fill out a questionnaire focusing on the experience of using either Strata or Unakite. We asked questions on the usability of the system they used in their respective conditions, the usefulness of such tables generated by the system, their opinions of the different features of the system, their willingness to author tables using the system to keep track of their decisions, their concerns about privacy if they were to author tables, as well as their familiarity with the topic of the three tasks used in the study. Finally, we ended the session with an informal interview on any additional thoughts they had about the system they used. Each study session took about 60 minutes per participant and was done remotely using the Zoom video-conferencing application. All participants were compensated $15 for their time.

6.6.2 Quantitative Results

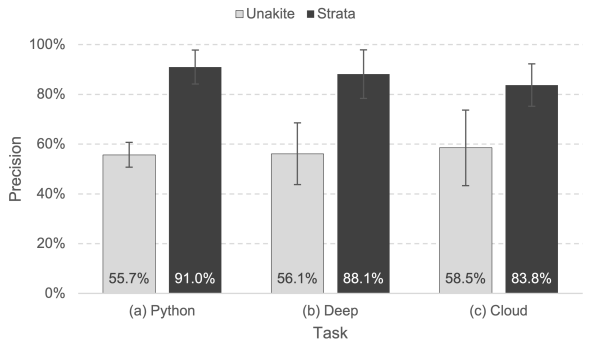
All participants were able to complete all of the tasks in both conditions, and none of them went over the pre-imposed time limit.

The results show that the participants in the Strata condition took significantly *less time* to finish compared to the Unakite condition for all three tasks, as shown in Table 6.2. Across all three tasks, the average time for completion was reduced by 32.5% when using Strata (Mean = 292.1 seconds, σ = 56.9 seconds) compared to using Unakite (Mean = 432.7 seconds, σ = 61.8 seconds), which is also statistically significantly ($p < 0.05$). Thus, using Strata did help participants evaluate the appropriateness for reuse faster.

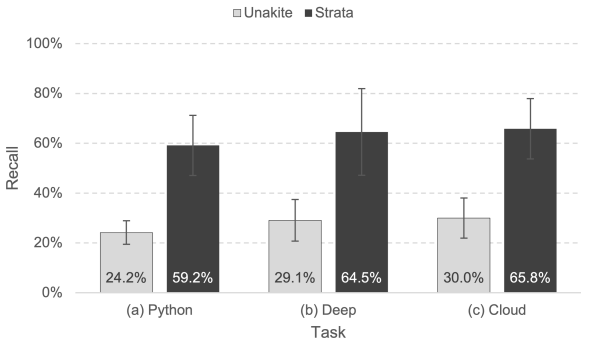
To assess the *quality* of the reasons that participants came up with, before the study, two professional developers who are not affiliated with the research each generated a list of *high quality* reasons for all three tables independently. After resolving conflicts through discussions between the two developers, we produced a list of high-quality reasons for each table as the "gold standard". We then calculated and report in Table 6.2 the numbers of high quality reasons participants identified that are on the "gold standard" list, as well as the precision (calculated as $n_{\textbf{High Quality}}/n_{\textbf{Total}}$) and recall (calculated as $n_{\textbf{High Quality}}/n_{\textbf{Ref. High Quality}}$) of high-quality reasons (where $n_{\textbf{Total}}$ is the total number of reasons they generated, and $n_{\textbf{Ref. High Quality}}$ is the number of "gold standard" high-quality reasons for each task). By plotting the precisions and recalls in Figure 6.6, we can see that participants in the Strata condition achieved higher precision in all three tasks, that is, they gave a higher percentage of high-quality reasons in their responses compared to the Unakite condition. Participants in the Strata condition also achieved higher recall in all three tasks, that is, they were able to find more high-quality reasons compared to the Unakite condition. Thus, using Strata did help participants improve the quality of their evaluations compared to using Unakite.

In case participants came up with valid answers we had not thought of, after the study, we asked the same two developers as above to rate each reason that participants gave as either *valid* or *not valid* blind to the conditions. Valid reasons are considered as the ones that are specific and correct according to the content of the table. After resolving conflicts through discussions between the two developers, we filtered out the reasons that are considered *invalid*, and presented the resulting numbers of valid reasons in Table 6.2 (the numbers of invalid reasons were negligible and were therefore not included in the table). Across all three tasks, the average total number of valid reasons ($n_{\textbf{Valid}}$) increased by 75.6% when using Strata (Mean = 7.90, σ = 1.90) compared to using Unakite (Mean = 4.50, σ = 0.94), which is also statistically significant ($p < 0.05$). Thus, using Strata appeared to help participants come up with more valid evaluations for appropriateness for reuse compared to Unakite alone.

In the survey, participants reported (in 7-point Likert scales) that they thought the interactions with Strata were understandable and clear (Mean = 6.20, Median = 6.00, 95% CIs = [5.75, 6.46]), they enjoyed Strata's features (Mean = 6.00, Median = 6.00, 95% CIs = [5.45, 6.72]), and would recommend Strata to friends and colleagues (Mean = 6.10, Median = 6.00, 95% CIs = [5.65, 6.35]).



(a) Precision of high quality reasons ($n_{\textbf{High Quality}}/n_{\textbf{Total}}$.)



(b) Recall of high quality reasons ($n_{\textbf{High Quality}}/n_{\textbf{Ref. High Quality}}$.)

Figure 6.6: Precisions and recalls of high quality answers in all three tasks. All results are statistically significant under t-tests ($p < 0.05$).

6.6.3 Qualitative Results

6.6.3.1 Usability and usefulness of Strata's features

Overall, participants appreciated the increased transparency and efficiency afforded by various Strata features and highlighted the values of the appropriateness properties that we visualize, arguing that *"it helps me understand how a table was made step by step"* (P10), *"lets me know what the author searched for, so if I don't understand something, I can search again. And more importantly, I can sort of know what the author didn't look for, and sometimes that'll become what I can do next"* (P4), *"[the automatic context snapshot feature] saves me lots of time that I would otherwise spend going to the source web pages and making sense of things, which could be a rabbit hole sometimes"* (P15), and *"[allows me to] see on a high-level where stuff comes from and if there's any source that is potentially questionable"* (P13). In addition, P8 reflected that Strata *"serve(d) as a guidance for things that I should pay attention to,"* which underlines the value of our framework, and reminded some participants of appropriateness properties that they would otherwise overlook, such as *"I never really thought about what the author(s) looked for or not, but now I think it's actually quite important, especially if they miss obvious things that an expert would never miss,"* (P6) and *"I realize that I'm more of a grab-and-go kinda person and I don't usually remember to check how many up-votes a Stack Overflow answer gets or when it was last updated"* (P17).

6.6.3.2 Authoring tables

Participants were also excited about authoring tables with Strata running, as it will automatically extract and produce the sidebar on the left and the various signals in the table. They mentioned that such *"honest signals enhanced"* (P10) tables would be particularly useful in situations such as code reviews (P6: *"going through the three main aspects is like going through our usual quality*

checklist, which makes sure that we're not missing anything") and project takeovers (P13: *"if my previous browsing sessions are captured by this, then I won't need to make myself available again and again if somebody else suddenly has a question that only I know the answer to, since I made it in the first place—this table thing will almost be self-explanatory"*).

6.6.3.3 Privacy concerns

Some participants shared their privacy concerns from an author's perspective, mentioning that certain types of metadata that could reveal their personal preferences and idiosyncrasies (e.g., the code that they used, the snippet surroundings, and their search queries) should be kept private until they felt comfortable sharing. Indeed, prior work has pointed out that there may be negative effects of surfacing certain types of information [106]. These findings identified new research opportunities for (1) intelligent mechanisms that can automatically screen for and block out information that should be kept private (e.g., similar to [221] or [177]) and (2) mixed-initiative and interactive mechanisms [164] that collaborate with users to only preserve the information that they are comfortable sharing (e.g., similar to [222]) without compromising the usability and effectiveness of the system.

6.7 Discussion

Prior research on web credibility stressed the importance of trustworthiness measurement during the evaluation of the appropriateness to reuse a previously created knowledge artifact [162]. However, as we found from literature on sensemaking handoff and our formative study, evaluating the appropriateness of reuse is much more than simply verifying the trustworthiness [162], especially since the artifacts are often an author's collection and synthesis of different individual pieces of information from different sources and reflect the author's opinion about the trade-offs among multiple valid options [232]. As a result, in addition to understanding whether the content is trustworthy, consumers also need to understand if the original problem context when the author created the artifact matches with the consumer's [162, 246], and if the author's research process was thorough [102, 283]. One of the contributions that we make in this work is a framework (Table 6.1) that summarizes the aforementioned three major facets, serving as a checklist that guides consumers through their evaluation processes. Strata, which is an instantiation of the framework, improves consumers' abilities to evaluate these facets compared to using Unakite alone, as evidenced by both the quantitative (i.e., number of valid reasons given by the participants

in terms of each facet) and qualitative results (e.g., participants' comments on Strata reminding them of double checking appropriateness properties that they would otherwise overlook).

Although prior work on trust and sensemaking handoff offers insights into the various aspects and properties that are important for evaluating the appropriateness of reuse, it remained costly and difficult for not only the author who was creating the knowledge to also keep track of those signals and save them somewhere (since it is extra work without immediate benefit), but also for the consumer who was interpreting the knowledge to deduce and speculate about those signals. Through our research, we learned that an reasonable number of appropriateness signals can automatically be captured at authoring time as well as processed and visualized to the consumers subsequently to help with the reuse evaluation, and thereby reduce the cost for people to build on each other's knowledge artifacts.

6.8 Limitations and Risks

There are certain types of information that Strata is not able to automatically obtain and visualize. One set of limitations results from Strata working in the browser, so it cannot monitor activities which happen in the authors' code editors or IDEs, command line interfaces, and relevant discussions with friends and colleagues (communicated either verbally or electronically through chat applications like Slack). Further development of extensions in these different environments as well as research into how to coordinate the collection and organization of this information would be needed in order to provide consumers with a more complete picture of an authors' working context beyond the browser. However, even in situations where Strata cannot automatically calculate a signal, we believe that the three major facets still alert consumers that these are important aspects to be considered. Also, to the extent that consumers come up with their own measurements and ways to fulfill their information needs, they are perfectly welcome to do so, such as testing if a piece of sample code returns the desired result by running it in a terminal, which the current Strata does not automatically do.

Some of the features in Strata are currently implemented based on heuristics, such as the bounds of the automatic context snapshots and the threshold beyond which information is considered out-of-date. These heuristics are based on our preliminary piloting through limited iterations, and may not apply universally to every situation. Further development can make these features more universally applicable and more adaptive to different situations so that users will be able to rely more on the judgments that Strata automatically generates.

The current design of Strata is intended for use cases where people collaborate and communicate their knowledge artifacts with each other in good faith; for example, software engineers sharing design rationale within a team. However, for Strata to be used at scale with potentially malicious actors, such as in situations where some authors might try to increase the trustworthiness and thoroughness scores by manipulating the different metrics that it uses and displays, additional signals as well as mitigation techniques might be needed to combat such gaming behaviors. One approach would be to aggregate multiple knowledge artifacts with similar context (options, criteria, and goals in the case of Unakite comparison tables) together and detect and filter out anomalous components, inspired by mechanisms like "down-voting" that community Q&A sites (e.g., Stack Overflow) use to guard against incorrect and malicious answers at scale. Further, some of the information, like the context, seems difficult and pointless to distort.

One of the concerns that repeated during our iterative design process is that each surfaced appropriateness property ultimately competes for user attention and takes time for the reader to process [194], which could result in the overall user interface being overwhelming. The current solution we employed, inspired by prior work in recursive summarization and sensemaking [378, 379], takes a hierarchical approach that presents users with an overview and the ability to dive into specific details, letting them take the initiative of exploring parts relevant to their own interests. Future research is needed to untangle the relative importance of the various factors and how they can be alternatively represented. One idea is to gather large amounts of usage data from a field deployment and develop statistical or machine learning-based models that can predict importance metrics given various input parameters.

Finally, our lab evaluation contains several limitations. Given the short amount of training time participants had, some may not have been able to get fully acquainted with the various features that Strata offers. The tasks used in the study may not be what participants encounter in their daily work, and participants may not have the necessary context and sufficient agency as they do in real life. We mitigated these risks by asking participants to complete a practice task simulating what they would need to do in the study to help them get familiarized with Strata as well as the flow and cadence of the tasks. To improve realism, all three tasks used in the study were based on actual questions asked by real developers online, and the tables used in the study were adapted by the first author from real comparison tables we found online. For each task, we also provided participants with some background information and context to get them prepared. In the future, we would like to further address these limitations by conducting a long-term larger-scale field study, where developers will have both sufficient familiarity with Strata through repeated usage and motivation to reuse decisions that are relevant to their own work.

6.9 Future Work

One participant (P4) in the evaluation study said *"I can imagine myself having this table page open as I collect stuff so I can check how well I'm doing as I go"*, which suggests that Strata not only can help consumers but also provides value for authors at collection time — authors can use Strata features to help them know how well their decisions will be judged, how thorough they have been, whether they are using up-to-date materials, if there have been any version mismatches, etc. Thus, future work can investigate how to integrate these Strata visualization features into authors' workflows to help them "proofread" their decision-making processes in real-time.

Currently, Strata has settings that consumers can tune based on their personal preferences, such as the trusted domain whitelist. Future work is needed to investigate mechanisms that can enable consumers to also personalize an existing table, such as adding, editing, and removing certain elements, effectively creating new versions of that table without overriding the original author's version. In addition, it would also be an interesting challenge to aggregate the changes in different consumers' versions and propagate them back to the original author as constructive feedback.

Finally, our approach may have potential implications for other situations and domains involving user-generated content (beyond comparison tables), in which knowledge consumers need to evaluate the relevance and trustworthiness of that content. For example, the context, trustworthiness, and thoroughness facets could provide generative inspirations for helping users evaluate how knowledge artifacts were constructed, such as in Wikipedia (e.g., which sources were considered for an article, properties of the contributors, and coverage of key topics mined from similar articles), Q&A sites like Stack Overflow where many people collaborate and edit questions and answers together; curation platforms such as Pinterest, or thousands of other wiki systems. Generalizing how to augment knowledge reuse for situations beyond decision-making in programming is an interesting and potentially fruitful area for future investigation, including exploring which information needs identified in this paper may not be as relevant and which additional needs become important. On the one hand, such an endeavor could unlock cycles of knowledge reuse in which people can quickly make good judgments about which information to aggregate and accumulate, which then become useful signals for making future judgments easier as well. On the other hand, the various signals and properties that are automatically surfaced could raise consumers' awareness of the potential existence of misinformation online [349] and provide readily available evidence to combat it.

Chapter 7

Selenite: Grounded Reading and Comprehension

Previous chapters in this thesis and past research have generally assumed that users have already discovered and comprehended the information they chose to integrate into the system. However, through formative interviews and inquiries with participants who recently explored unfamiliar topics and domains, we discovered that people frequently grapple with finding, reading, understanding, and navigating information in the first place, largely due to a lack of comprehensive overview of the information space. This deficiency arguably hinders the subsequent development and refinement of their mental models.

Therefore, in this chapter, we introduce a novel system named Selenite that provides users with a comprehensive overview of the information space upfront to jumpstart as well as guide their subsequent reading and sensemaking processes. Through a performance evaluation of Selenite, we verified its ability to provide an accurate and high-quality global overview to the users. Furthermore, two user studies revealed that Selenite significantly accelerated users' information processing, improved their comprehension, and effectively facilitated the discovery of relevant and diverse information.

7.1 Introduction

Whether it is parents delving into the vast sea of baby stroller choices or developers choosing a JavaScript frontend framework, people frequently find themselves having to navigate through unfamiliar topics and domains. One key challenge here is that people lack a comprehensive understanding of the information space when they start [143, 193]. For example, first-time parents

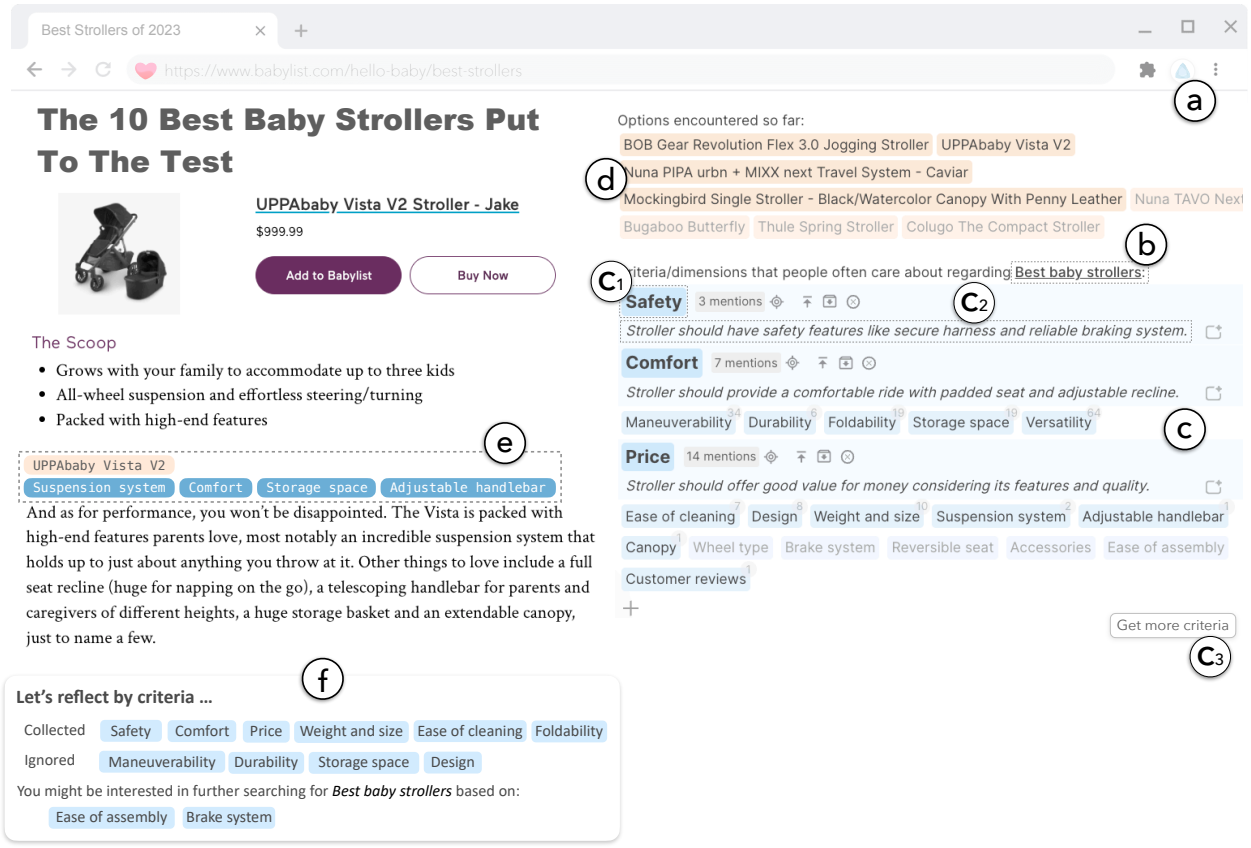


Figure 7.1: Main user interface of Selenite, which provides users with a comprehensive overview of the information space in the sidebar (a). When users encounter an unfamiliar topic (b), Selenite offers them a global grounding based on commonly considered criteria (c) as well as the options encountered so far (d), helping them develop quick intuitions of the topic. As users read new articles, Selenite provides local grounding through page-level and paragraph-level summaries and annotations (e), enabling effective comprehension and efficient navigation between the content of their interests. Upon leaving a page, Selenite dynamically summarizes users' progress and suggests avenues for finding additional new information (f) in subsequent searches.

might not recognize important parameters for baby strollers like maneuverability, adaptability for baby growth, or compatibility with car seats, consequently narrowing their selections. Similarly, novice developers might not be aware of crucial factors of a frontend library like stability, community size and support, or ease of integration with existing codebases, resulting in the choice of a sub-optimal library.

Currently, the process of coming to understand an information space is messy, requiring individuals to find and read various information online [71,192,193,204], break down often verbose and circumlocutory content into more digestible parts [72,80,159,235], and iteratively collect, organize, and refactor this information to make decisions [232,247]. Prior research [37,145,181,315], including the work described in previous chapters of this thesis [232, 234, 235], has concentrated predominantly on optimizing the process of collecting and organizing information, i.e., the *final stages* of sensemaking [288, 307]. However, people frequently feel overwhelmed when trying to

find and read information *to begin with*, particularly when they lack a comprehensive overview of the information space initially [143]. One challenge lies within the reading process itself — people may struggle to fully comprehend and absorb certain content, and may also fail to recognize important aspects that would warrant their attention [196, 289], leading to a limited viewpoint and misguided decisions [147,370]. For example, one participant from our formative study shared her experience of reading baby strollers reviews. She confessed that "*I never had a baby or even cared for one before, so the part about how adjustable a particular stroller was didn't make sense to me immediately. It wasn't until later, when an experienced mother mentioned to me that a baby can grow pretty fast, and it's important for a stroller to be able to keep up and adapt, did I realize that it's actually a very important aspect.*"

Another challenge arises when people have to seek and sift through numerous online reviews and comparison articles but with limited time and cognitive bandwidth — a more complete understanding of the information space ideally requires reading "everything," but this is usually impractical or impossible. Instead, one popular practice that people adopt is "selective (or non-linear) reading" [362], where they only read the paragraphs that discuss information that they consider relevant or valuable and bypass the rest [89,361], e.g., in a session where they would like to compare different options with respect to a particular criterion. However, it is challenging for people to gauge the potential value of articles or paragraphs, especially long-winded ones, just by skimming, without delving into a more thorough read [60,112,118]. For instance, information snippets about the maneuverability of different baby strollers may be dispersed throughout a review and appear in diverse variations (e.g., "agile enough to go through tight spots" and "easy to steer and navigate small corners" both arguably discuss "maneuverability"), making it difficult for people to spot these variations effectively and navigate efficiently among such scattered details. Furthermore, after finishing reading a page, people often want fresh insights that could broaden their perspectives rather than remaining confined within their current sphere of knowledge [33, 147]. However, they often encounter a challenge wherein multiple articles about the same subject may contain information that largely overlaps [154, 190, 249]. In these scenarios, people may struggle to formulate effective search strategies that would reliably guide them to subsequent pages that contain novel content, maximizing the information gain from each reading.

In a formative needs-finding study, we found that people expressed a desire for a comprehensive overview while reading and digesting unfamiliar topics, e.g., receiving an "expert summary" of the crucial factors before diving into the actual content. In addition, they found that it is currently inefficient and ineffective for them to manually seek and locate desired information

snippets (e.g., "*everything about the maneuverability of a stroller*") both within the current page of interest as well as from subsequent search engine result pages (SERP).

To overcome these challenges with finding and reading information, we explore the idea of providing users with a comprehensive overview of the information space upfront to jumpstart as well as guide their subsequent reading and sensemaking processes in a novel system named Selenite.[1] When users encounter an unfamiliar topic, Selenite leverages `GPT-4` (a large language model developed by OpenAI) as a knowledge retriever to offer them a global grounding based on commonly considered criteria, helping users develop quick intuitions of the topic. As users read new articles, Selenite provides local grounding through page-level and paragraph-level summaries and annotations, enabling effective comprehension and efficient navigation of the content of their interests. Upon leaving a page, Selenite dynamically summarizes users' progress and suggests avenues for finding additional information to expand their perspectives rather than duplicating existing knowledge. Through a performance evaluation of Selenite, we verified its feasibility to provide a sufficiently accurate and high-quality global overview to the users. Furthermore, usability and case studies revealed that Selenite significantly accelerated users' information processing, improved their comprehension, and effectively facilitated the discovery of relevant and diverse information.

The contributions described in this work include:

- a formative study showing people's need for support in terms of finding and reading information despite recent advances in information collection and organization tools,
- Selenite, a novel system providing users with a comprehensive overview of the information space upfront to jumpstart as well as guide their subsequent reading and sensemaking processes,
- a performance evaluation of Selenite that demonstrates the feasibility of our approach,
- usability and case studies of Selenite that offer insights into its usability, usefulness, and effectiveness.

7.2 Formative Study & Design Goals

To better understand the obstacles people encounter in their information-seeking and reading strategies during sensemaking, we conducted a formative study. Building upon the insights from this study and existing research, we established a clear set of design goals.

[1] Selenite is named after a soft and transparent gemstone, and stands for "**S**mart **E**nvironment for **L**ogical **E**xtraction and **N**avigation of **I**nformation using **T**echnological **E**xpertise."

7.2.1 Formative Study

7.2.1.1 Methodology

Participants were a convenience sample of 8 information workers (5 male, 3 female) recruited through social media listings and mailing lists. To capture a variety of processes, we recruited 3 doctoral students, 2 professional software developers, 2 researchers, and 1 administrative staff member. While we do not claim that this sample is representative of all information workers, the interviews were very informative and helped motivate the design of Selenite.

We began by asking how frequently participants conduct learning and comparison tasks on topics that were not familiar with that involved a lot of information-seeking and reading (we subsequently kept track of these topics and used them in our system evaluations). We then explored how they manage these situations. We asked the participants to provide context by reviewing their browser histories to cue their recollections while retrospectively describing recent sense-making tasks, projects, or problems. We solicited their workflows, strategies, mental models, frustrations, and needs. Finally, we had participants use our previous Unakite system to make sense of a topic that they are not familiar with (e.g., for people who have not yet had children to figure out the best baby strollers to purchase for their future child) and gathered their workflows, strategies, mental models, as well as feedback, opinions, frustrations, and needs. We used Unakite since it was shown to be easy to learn and use, and can support virtually all webpage styles and structures (as opposed to Crystalline, which only support a limited set of webpages in the domain of programming)

7.2.1.2 Findings

Table 7.1 summarizes all the unfamiliar topics (a total of 28 topics) that participants reported encountering and exploring. Below, we report major findings from the study:

People often find themselves feeling lost or unsure of where to begin and desire a big-picture understanding of important criteria (or aspects) of an information space before diving deeper. When approaching an unfamiliar topic, one common strategy that participants reported employing is to find some sort of "*overview of different aspects*" (P3) that would give them "*an intuition of what to care about and some guidance on what to look out for for each option*" (P5) in their subsequent exploration. For example, when investigating which time-tracking app to use, P5 was able to find a few articles that provided such overviews at the beginning, e.g., under the section "*What makes the best time tracking software?*"[2], which lists "*real-time tracking*", "*the*

[2] `https://zapier.com/blog/best-time-tracking-apps/`

Index	Theme	Topic	Participants
1	Software & online services	Choosing a hybrid app framework	P2, P5
2		Selecting a secure password manager	P3, P7
3		Choosing a suitable ERP (Enterprise Resource Planning) solution	P1
4		Choosing a reliable VPN (Virtual Private Network) provider	P1, P5
5		Picking a deep learning framework	P8
6		Deciding on the best data visualization tool	P6
7		Choosing the best time tracking tool	P4
8	Consumer Electronics & Technology	Choosing a high-quality digital camera	P2, P5
9		Choosing the best action camera	P8
10		Selecting a VR headset	P7
11		Picking a drone	P3, P5
12		Picking a smart home ecosystem	P1, P6, P8
13	Home Appliances & Furniture	Picking the best robot vacuum	P2, P5
14		Choosing the best air purifier	P4
15		Selecting the best washing machine	P3
16		Picking the right refrigerator	P3
17		Selecting the best mattress	P3, P6
18	Outdoor & Adventure	Choosing the best city bike	P7
19		Choosing the best barbecue grill	P3
20		Choosing the best tropical vacation location	P4
21	Health & fitness	Choosing an effective diet plan	P1, P7
22		Picking a reliable treadmill	P3
23		Picking the best running shoes	P8
24	Gifts & special events	Choosing a birthday gift	P5, P6
25		Picking the right wedding venue	P2
26		Picking the perfect engagement ring	P2
27	Parenting	Choosing the best baby stroller	P8
28	Pets	Choosing a breed of dog to adopt	P4

Table 7.1: Unfamiliar topics (organized by themes) that participants in the formative study reported encountering and exploring. Some topics were explored by multiple participants, such as "Picking a smart home ecosystem" and "Choosing a reliable VPN provider."

ability to edit time tracked or manually add time blocks", "*reporting features*", "*the ability to create an invoice or export data*", and "*multiple points of access*" as the most important criteria. However, participants complained that such lists of criteria are often "*subjective, incomplete*" (P1), contain aspects that they "*most likely don't care about*" (P2), and worse yet, "*do not represent how the rest of an article would be structured*" (P6). In addition, for certain topics such as "*best birthday gift ideas*", such overviews of criteria are hard to find up-front, in which case they would have to employ a bottom-up approach by reading through a series of articles back-and-forth, which is often considered "*time-consuming*" (P1) and "*hard to actually follow through*" (P2). Without these "*important criteria to keep in mind*" (P4) up-front, participants reported feeling "*overwhelmed by large amounts of unfamiliar information*" (P6), lacking "*a sense of clarity and structure*" (P7), and can easily lose focus during sensemaking. These findings prompted us to generate an initial overview of the commonly considered criteria given an information space to provide users with some global grounding and an anchor point for their subsequent reading and sensemaking.

Identifying and consistently keeping track of criteria is challenging, even with lightweight collection and organization mechanisms introduced in Unakite. Participants reported struggling with identifying and unifying criteria while reading content, which is a "*significant cognitive load*" (P4) when they are unfamiliar with a topic. One of the challenges is that the same criterion can be discussed in various ways across different articles (and even within the same article), and it is hard for participants to recognize those variations and time-consuming to flip back and forth to make sure they are unified and consistently represented in their Unakite table. For example, when investigating the topic of "baby strollers," P6 first saw a stroller should be "agile and nimble to be able to go through tight spots and sidewalks," and put it down as "nimbleness" in Unakite; later when she saw another segment that stated that a particular stroller is "easy to steer and handle and can smoothly navigate tight corners," she created another criterion called "steering." It wasn't until when she saw a segment in a third article that described a stroller having great "maneuverability and control" did she realize that all of these were practically describing the same aspect, "maneuverability," and she had to go back and readjust and combine those criteria and their associated evidence in Unakite. Additionally, P7 recounted a similar experience when searching for washers and dryers for his first house and admitted that "*oftentimes, coming up with the right keyword or jargon to summarize what I saw can be surprisingly hard, and I really wish someone would just do that for me.*" Selenite tackles this issue by providing a comprehensive list of frequently considered criteria upfront, eliminating the need for individuals to find the criteria themselves.

On the other hand, participants also reported misinterpreting content in their first passes. For example, P1, who tried to pick a natural language processing (NLP) online course to take, first thought things such as "sentiment analysis," " question answering," and "semantic role labeling" were different "aspects or branches" of NLP, but later realized from talking to a fellow researcher that they were all "tasks" that are used to evaluate NLP models. In these situations, participants reflected that 1) it was difficult to realize their misinterpretations without "*expert advise*" (P2), and 2) it was hard to retroactively locate and revise their previous categorizations of information (if they externalized them in the first place, e.g., using note-taking tools like Google Doc or Unakite).

People need reading and navigation guidance both at paragraph level as well as article level. Participants reported often having "*limited attention span*" (P8) when reading online articles and can only focus on a certain amount of information, usually the first few paragraphs or the first few sentences within a paragraph, before getting distracted or lost. For example, P5 in her quest to find a suitable time-tracking app pointed to a typical situation where "*sometimes a paragraph, even a short one, could be quite convoluted and have a lot of intertwined information,*

for example, and at first I thought this paragraph was just about money, but the rest of the paragraph was actually about lots of other things like compatibility." But, since participants tend to skim through content quickly, it often leads to potential misunderstandings or missing important details. In such situations, participants desired "*some simple metadata of what's covered in a paragraph*" (P4) to give them an intuition of what the paragraph is about and whether it is worth reading. These findings prompted us to provide clear in-situ per-paragraph summaries and the option for users to clarify convoluted paragraphs by "zooming in" on them in Selenite.

The same applies to the page level, where participants wanted to be able to "*preview a page before investing time reading it*" (P3) to understand whether it discusses detailed aspects that they care about. Additionally, such preview can also help them "*maximize the information gain from each page*" (P5), i.e., help them avoid reading duplicate information and aspects without learning anything new. As P4 put it, "*if I've already learned about all the aspects from the other pages, I don't have to read this one.*" However, as discussed previously, such previews (even if they are in the form of an abstract or table of contents) are not always available for each article. And most importantly, even if they do exist, they are almost always not grounded by a person's past reading and information collection activity. Selenite fulfills the page-preview need by offering users a concise overview of what's covered (or not) in a page to help users gauge its value. Additionally, Selenite tackles the issue of personalization by presenting users with a progress summary based on their previous sensemaking activities upon completing a page.

Furthermore, as participants became more familiar with a topic, their reading patterns started to get increasingly selective and non-linear. For example, we have observed that participants use a combination of keyword searches and flipping back and forth in an effort to find relevant information about a particular criterion that they cared about (with respect to different options), which they thought was "*haphazard*" (P1) and "*inefficient*" (P7). This led us to suggest potentially fruitful search keywords to users for discovering more unseen information in the end-of-page progress summary.

7.2.2 Summary of Design Goals

We postulate that an effective user interface/interaction paradigm for helping users find and read about key information during sensemaking should support:

- **[D1]** As the user enters a webpage, **provide a *global grounding* using common criteria as well as the options encountered** to help users build intuitions of the information space and promote structured thinking;

- **[D2]** During their reading, **provide a *local grounding* using page-level as well as paragraph-level summaries and annotations** to enable an accurate understanding of and effective navigation within and across articles;
- **[D3]** Upon finishing reading a page, **dynamically suggest next steps in sensemaking based on users' existing reading and information collection activities** to avoid missing important aspects *after reading* as well as maximize the information gain *in future reading*.

7.3 System

Based on the design goals, we designed and implemented the Selenite Chrome extension prototype to help people read about and make sense of unfamiliar topics with the help of global grounding. We will first illustrate how an end-user, Adam, would interact with Selenite (summarized in Figure 7.2).

7.3.1 Example Usage Scenario

Adam, an expectant father, is seeking guidance in selecting a baby stroller for his upcoming child. As someone without prior experience in child-rearing, he decided to rely on Selenite to help him while going through review articles and product pages of baby strollers.

Adam did a quick Google search and clicked on the first result page, which appeared to be a review article titled "The 10 Best Baby Strollers Put To The Test". Upon opening the page, Selenite automatically recognized the **topic** of the page as "best baby strollers" (Figure 7.1b), and then automatically presented a global overview in the sidebar that is injected directly into every web page (Figure 7.1a). The overview contained a list of **criteria** (Figure 7.1c) that are commonly considered by people when discussing the topic (Figure 7.1b), in this case, "best baby strollers." In addition, Selenite also automatically parsed the web page content and extracted the different baby stroller **options** and presented them under the "Options encountered so far" section (Figure 7.1d) in the sidebar. Notice that, without any additional effort from Adam, he received the automatic recognition of topics, retrieval of commonly considered criteria, and extraction of options from the page "for free," all of which occurred within roughly 5 seconds. This overview provides Adam with a comprehensive grounding of the information space regarding the topic of "best baby strollers." After quickly skimming them, Adam now felt that he already has an intuition built up

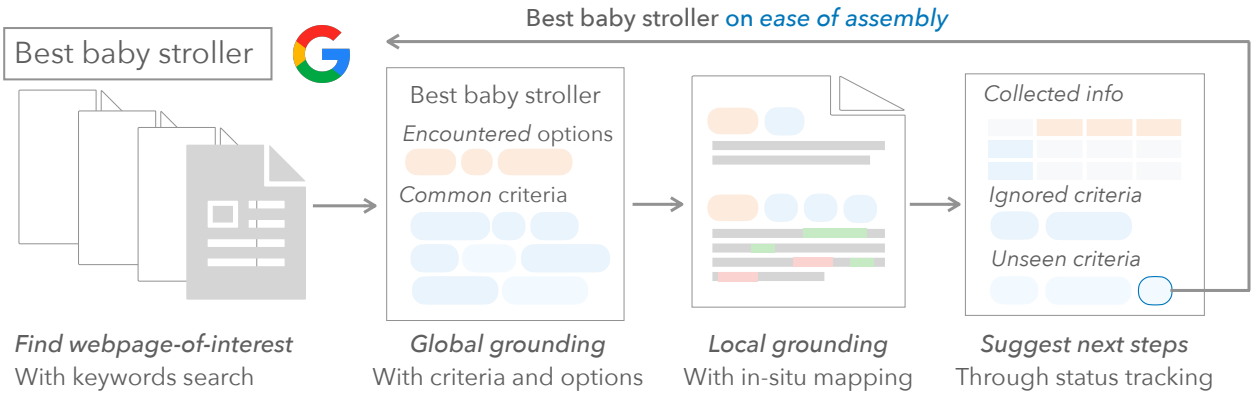


Figure 7.2: Main stages and features of Selenite: After finding an initial webpage-of-interest to read, Selenite provides 1) global grounding with a set of common criteria as well as options encountered so far, 2) local grounding with in-situ mapping of criteria per paragraph, and 3) suggestions for next steps in sensemaking.

about what criteria he should look out for when picking baby strollers, all without delving into the article itself.

Based on the global options and criteria, Selenite highlights and *contextualizes* the *local ones* that exist on the current page (by highlighting them in the sidebar (also with the ones that are not present on the page low-lighted, for example, see Figure 7.1c&d), helping users better understand and find specific information of interest while browsing. For example, as Adam read the article, he noticed that Selenite provides **in-context annotations** of **mentioned criteria** above each paragraph (which research has shown to be the unit of information that people usually think in and work with during sensemaking [247, 314]) that is on the page (Figure 7.1e). He quickly learned that he could just skim those mentioned criteria to get a rough idea of what a particular paragraph is about and decide if that paragraph is worth reading.

When he came across information about the *maneuverability* of a specific stroller while reading the article, Adam became interested in finding out if there were any details about the maneuverability of other stroller options as well. To facilitate this, he decided to use the "locate previous/next" buttons (Figure 7.3a) to quickly navigate among the paragraphs that discussed maneuverability. Here, the aforementioned annotations not only offer paragraph overviews during *linear* skimming but also act as bookmarks for *non-linear* navigation between distinct parts of the page that pertain to similar criteria.

Later, when Adam encountered a particularly convoluted paragraph with multiple criteria and options that he couldn't quite absorb in the first pass, he decided to leverage the "zoom in" feature that Selenite offers — he can query for more comprehensive descriptions that clarify which sentences or phrases within the paragraph pertain to specific options, criteria, and sentiments

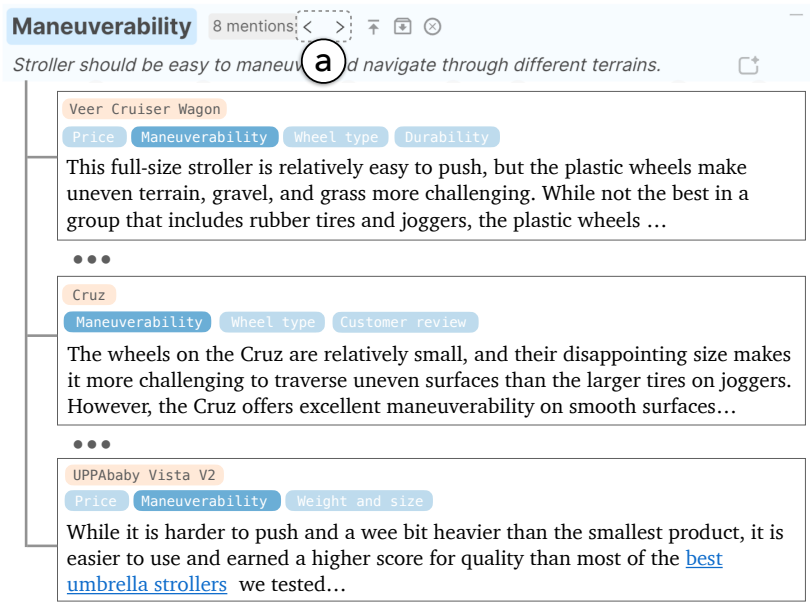


Figure 7.3: Selenite enables structured and effortless navigation by selected criterion through clicking the "locate previous/next" buttons (a).

(positive, neutral, or negative) (Figure 7.4) by clicking the "Analyze" button (Figure 7.4a) that appears when hovering the cursor over a paragraph.

As Adam navigated the page and read the content, Selenite leveraged the implicit behavior tracking capabilities of Crystalline (described in chapter 4, section 4.3.2.2) to keep track of the criteria that he paid attention to on the page. Additionally, when Adam encountered particular paragraphs that he wanted to keep track of in case of revisitation, he used the wiggling gesture (described in chapter 5, section 5.4.1) to collect that paragraph into a repository (similar to the one in Unakite).

When Adam reached the end of the current article, Selenite presented a **summary** block (Figure 7.1f), automatically describing his research status and recommending possible next steps. This summary contains three sections: (1) criteria and options users have seen evidence for based on implicit and explicit behavior tracking (emphasizing their focus and priorities), (2) the remaining ones that occurred on the page (to help users confirm that they skipped certain information intentionally, not by oversight), and (3) a set of suggested search queries that can potentially help users find subsequent web pages for broadening their perspectives and maximizing their information gain. As suggested by Selenite, Adam then searched for "baby strollers that are easy to assemble" and "brake system of baby strollers" on Google, finding additional articles that contain information about these previously not encountered criteria.

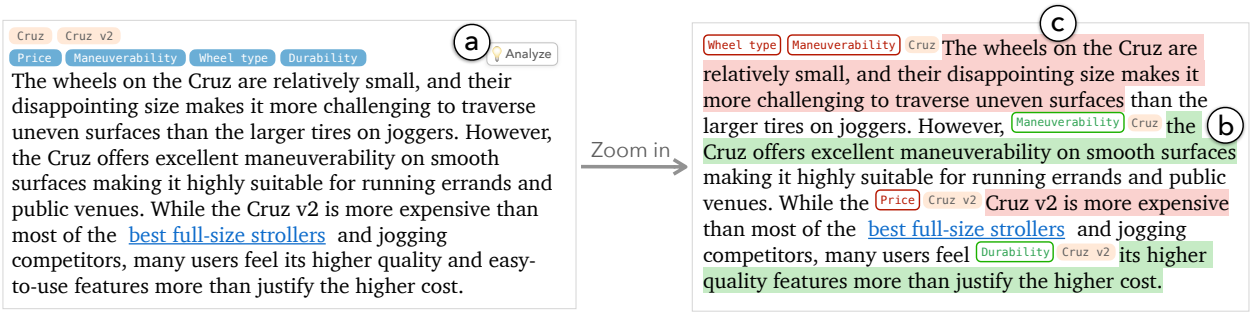


Figure 7.4: When encountering a particularly convoluted paragraph (e.g., the paragraph on the left) with multiple criteria and options that a user couldn't quite absorb in the first pass, they can leverage the "zoom in" feature that Selenite offers — they can query Selenite for more comprehensive descriptions that clarify which sentences or phrases pertain to specific options, criteria, and sentiments. Selenite wraps phrases and sentences in colored boxes, with green denoting "positive" (b), red denoting "negative", and grey denoting "neutral" (not shown).

7.3.2 Detailed Designs

We now discuss how the different features in Selenite are designed and implemented and how they support our design goals.

7.3.2.1 [D1] Providing Global Grounding using Common Criteria and Options Encountered

In Selenite, we explore the idea of having the system provide users with an initial overview of criteria that are typically significant and frequently considered by people when exploring a particular topic. By doing so, we aim to jumpstart their subsequent sensemaking processes. This approach offers multiple advantages: it not only provides users with a comprehensive and contextual understanding of the topic [66] but also serves as a gentle reminder of diverse perspectives and aspects that might have otherwise escaped their consideration. Additionally, these criteria act as a grounding for users, enabling them to process information from web pages and make meaningful connections more effectively and efficiently. Selenite also performs information extraction on each page to identify the options that a user has encountered during their sensemaking process. Naturally, users have the flexibility to reorder, pin, edit, add, or delete any options and criteria to tailor them precisely to their specific preferences. We discuss the relevant designs and the rationale behind those designs below:

Automatically recognizing topics. Selenite goes beyond previous sensemaking systems (e.g., Unakite [232], Crystalline [234], Wigglite [235], Fuse [204], Tabs.do [75], etc.) by autonomously identifying and classifying web pages into broad topics based on their titles and content. For

example, "React vs. Svelte: Performance, DX, and more"[3], "Angular vs React vs Vue: Which Framework to Choose"[4], and "What are the key differences between Meteor, Ember.js and Backbone.js?"[5] would all be recognized as "Comparison of JavaScript frameworks"; while "iRobot vs Shark Vacuums Bought, Tested, and Compared"[6] and "The best robot vacuum you can buy right now"[7] would go under the topic of "best robot vacuums." Unlike previous systems that require users to manually create projects or folders, Selenite further lowers the barrier for entry, enabling users to quickly begin utilizing and reaping the benefits of the system, especially from the list of commonly considered criteria based on that topic.

To achieve this, we frame the topic recognition as a *summarization* task for a large language model, i.e., `GPT-4`[8], where the input context is the title and initial five paragraphs of a given web page. Specifically, we first asked `GPT-4` (taking advantage of its generalizability to various domains [274], see section A.1 for the detailed prompt design) to summarize the web page given its title and initial paragraphs (with the temperature set to 0 to minimize LLM generation variability)[9] and then offer a search phrase that would enable us to find similar web pages using a modern search engine, which we use as the topic. Selenite then clusters the semantically similar topics (note that each web page has an associated topic generated by `GPT-4`) based on the cosine distances on topic semantic embeddings computed using SentenceBERT [296], as shown in the previous paragraph. Naturally, users have the flexibility to manually create, edit, and remove topics, as well as reassign pages to different topics based on their personal opinions.

Automatically retrieving commonly considered criteria. If we adopt the "bottom-up" approach discussed in previous chapters, an intuitive method for obtaining criteria would involve extracting them from individual paragraphs on a page. However, in our initial attempts, we found that this method faced significant challenges that limited its effectiveness. One of the main issues was the lack of uniformity among the criteria extracted from different paragraphs — each paragraph presented its own variations and nuances, making it difficult to establish a cohesive and standardized set of criteria (similar to what was reported by the formative study participants). Additionally, the approach lacked a comprehensive global perspective, failing to consider the broader context and overarching themes of the topic. As a result, manual review, correction, and

[3] `https://blog.logrocket.com/react-vs-svelte/`

[4] `https://www.codeinwp.com/blog/angular-vs-vue-vs-react/`

[5] `https://backbone447.rssing.com/chan-17153281/article10.html`

[6] `https://www.rtings.com/vacuum/learn/irobot-vs-shark`

[7] `https://www.theverge.com/22997597/best-robot-vacuum-cleaner`

[8] We used the `GPT-4` APIs provided by OpenAI to communicate with the underlying language model

[9] Empirically, we found this step helped `GPT-4` to better engage with the context provided. It also aligns with the idea of Chain-of-thought prompting proposed by [356].

	Criteria Name	Criteria Description
1	Safety	Ensuring the stroller has proper safety features such as a secure harness, sturdy construction, and reliable brakes.
2	Comfort	Providing a comfortable seat with adequate padding and support for the baby, as well as adjustable recline positions.
3	Maneuverability	Having smooth and easy maneuverability, with features like swivel wheels, suspension systems, and the ability to navigate tight spaces.
4	Durability	Ensuring the stroller is built to last, with high-quality materials and strong construction.
5	Storage	Offering ample storage space for carrying essentials such as diaper bags, snacks, and personal items.
6	Folding and Portability	Allowing for easy folding and compact storage, as well as being lightweight for convenient transportation.
7	Versatility	Providing features that allow the stroller to adapt to different terrains, weather conditions, and age ranges.
8	Ease of Use	Having user-friendly features like adjustable handles, intuitive controls, and easy-to-clean fabrics.
9	Price	Considering the affordability and value for money in relation to the features and quality of the stroller.
10	Customer Reviews	Taking into account feedback and recommendations from other parents who have used the stroller.
11	Weight and size	Considering the weight and size of the stroller to ensure it is manageable and fits well in different environments.
12	Ease of cleaning	Ensuring the stroller is easy to clean and maintain, with removable and washable fabric components.
13	Adjustability	The stroller should have adjustable handlebars and footrests to accommodate different caregivers and growing babies.
14	Canopy	A large and adjustable canopy to protect the baby from the sun and other elements.
15	Reversible seat	Having the option to face the baby towards the parent or away from the parent.
16	Brake system	Having a reliable brake system that is easy to engage and disengage.
17	Compatibility with car seats	Offering the ability to attach a car seat to the stroller for convenient travel.
18	Adjustable height	Allowing for adjustable handlebars to accommodate different heights of caregivers.
19	Easy assembly	Providing clear instructions and easy assembly process for the stroller.
20	Design and aesthetics	Considering the overall design and aesthetics of the stroller to match personal preferences.
21	Weight capacity	Specifying the maximum weight limit the stroller can safely carry.
22	Warranty	Checking for a warranty or guarantee that covers any potential defects or issues with the stroller.
23	Brand reputation	Considering the reputation and reliability of the brand manufacturing the stroller.
24	Accessories	Offering additional accessories such as rain covers, mosquito nets, or parent organizers for added convenience.

Table 7.2: The list of commonly considered criteria that Selenite retrieves for the topic of "best baby strollers" by leveraging `GPT-4` as a knowledge retriever.

unification of the extraction results were frequently necessary, making the process impractical and inefficient.

To address these challenges, we instead explored an alternative "top-down" approach, where we directly query an "oracle" for a globally applicable and comprehensive set of criteria. We are particularly inspired by recent research indicating that *the majority of people's information-seeking needs are not novel* [237] — "previous people" have experimented with most search needs and synthesized information into summarized knowledge such as review articles. While it is impractical for individuals to process and synthesize vast amounts of information online, LLMs excel at this. Recent studies suggest that LLMs can be highly effective in processing and integrating information, making them potentially valuable for tasks like knowledge graph querying and retrieving common sense knowledge [28, 352, 369], and, in our particular case, a suitable "oracle" for providing a set of commonly considered criteria given a particular topic.

In Selenite, we take advantage of the fact the LLMs have aggregated various review articles during its pre-training, and use `GPT-4` as a *knowledge retriever* — for any given topic, we prompt

it to produce a list of around 20 commonly considered criteria (Figure 7.1c), complete with their respective names (Figure 7.1c1) and descriptions (Figure 7.1c2) for each topic.[10] We strive to minimize potential anchoring bias by achieving a balance between relevance and diversity in our prompting strategy. On the one hand, we specifically requested criteria that are deemed as "most relevant to the topic," "frequently considered," and can "cover a broad range of perspectives." On the other hand, we adopted the Self-Refine technique [241], employing an iterative query approach with `GPT-4`. In each iteration, we requested the generation of five additional criteria that were "different, more diverse, and more important" than the previous ones (see section A.3 for the detailed prompt design). We also relied on `GPT-4` for ranking the criteria based on their importance. The whole generation process remains within a reasonable time frame (approximately 5 to 10 seconds, depending on the topic), which we deemed sufficient to test the idea of supplying commonly considered criteria as a form of global grounding to users' reading and sensemaking processes. Still, users have the freedom to request additional criteria (without repetition) if they believe the existing list is not comprehensive enough (Figure 7.1c3). Here, we document the list of 22 criteria that Selenite retrieves for the topic of "best baby strollers" in Table 7.2 to offer an intuition of the quality and coverage of our current `GPT-4`-based approach. We also present a performance evaluation in section 7.4 that provides initial evidence that this approach is sufficient for our prototyping purpose. We leave for future work to experiment with advanced approaches, such as retrieval-augmented LLMs [215], that would potentially provide increased perceived external validity.

Automatically recognizing encountered options. Instead of relying on `GPT-4` to access its internal knowledge and retrieves a set of commonly considered options, we instead leverage its zero-shot *information extraction* capability[11] and expansive context window size [274][12] to directly extract options from the entire text content of a web page. This approach ensures that the options presented in the sidebar align with a user's sensemaking process, i.e., they are indeed what users have encountered as opposed to something that users would potentially never run into. It also circumvents the potential concern where the world knowledge of an LLM is out-

[10] While this is out of our scope, we note that this criteria ideation approach can generalize to more novel or emerging topics, as LLMs can use the latest information on the Internet as context if paired with retrieval modules (essentially what powers Bing Chat [257], which, at the time of this writing, has no publicly accessible API yet).

[11] Zero-shot capability refers to the ability of an LLM to perform tasks with only instructions but no input-output examples.

[12] The context window of large language models is the range of tokens the model can consider when generating responses to prompts.

of-date, for example, ChatGPT only "knows" information up to September 2021 [57, 276].[13] In addition, it surpasses the limited heuristics employed in previous approaches such as Crystalline, which rely on page titles and HTML <h>-tags as sources for options. This is crucial because studies have consistently demonstrated that web developers frequently disregard semantic web standards and best practices [155,250]. For instance, it is common to find pages where every piece of content is enclosed in tags regardless of their semantic roles.

To enhance the reliability of the GPT-4-based option extraction pipeline, we have found it useful to follow these steps[14]: firstly, we prompt GPT-4 to determine if a page (based on its title and content) is likely discussing multiple options, such as product review and comparison pages or blog posts, or focusing on a single option, like item detail pages on platforms like Amazon or Airbnb. Once determined, we instruct GPT-4 to extract the relevant options accordingly (see section A.2 for the detailed prompt design). Interestingly, it appears that GPT-3.5 [276] lacks the necessary reasoning capabilities for this particular option extraction task.

7.3.2.2 [D2] Providing Local Grounding using Page & Paragraph-level Summary and Annotation

As the Internet continues to overflow with information, not all of it holds equal value for individual users who face the challenge of limited attention bandwidth. To navigate this deluge, many people resort to skimming content swiftly [118, 238, 338]. Yet, our formative study uncovered a major drawback to skimming – convoluted and intertwined paragraph and page structures often result in overlooked information or an inability for users to comprehend information. To address these issues, Selenite offers the following features:

In-context summaries and annotations of paragraphs. With access to the initial set of common criteria as well as the options extracted from each page, Selenite performs content analysis on each paragraph within a given page to identify the specific criteria being discussed and presents them as in-context annotations above the respective paragraph (Figure 7.1e). This feature enables users to swiftly scan through a page, understand the key points of each paragraph, and selectively concentrate on the paragraphs that are valuable and engaging for gathering information.

[13] While the direct retrieval of criteria from LLMs may also face this potential issue, in practice, we operate under the assumption that criteria are unlikely to suddenly emerge or become outdated.

[14] Empirically, we found these steps to be useful for our specific information extraction task using a large language model.

Such content analysis is enabled by recent advances in NLP, specifically large pre-trained transformer models [99, 214, 345] fine-tuned to perform zero-shot sequence classification tasks following a natural language inference (NLI) paradigm [373]. Specifically, for example, to assess if a given text (e.g., "`Angular is very hard to pick up`") covers the criterion of "`learning curve`," we can input the text as the *premise* and a *hypothesis* of "`This content discusses {learning curve}.`" into the NLI model. The entailment and contradiction probabilities are then converted into label probabilities, indicating the likelihood that the content pertains to the specified criterion. We used the `bart-large-mnli` model[15] for this purpose and considered options and criteria with a score above 0.96 as true positives, displaying them in descending order of scores. We determined this threshold empirically, prioritizing recall over precision, as discussed further in section 7.4.

In scenarios where users still struggle to comprehend content despite the presence of in-context annotations, Selenite can perform a deeper analysis on-demand by leveraging the advanced reasoning capabilities of `GPT-4`. Specifically, through parallel and carefully-orchestrated prompts, Selenite produces a more comprehensive description that clarifies which sentences or phrases pertain to specific options, criteria, and sentiments (positive, neutral, or negative) (Figure 7.4, also see section A.4 for the detailed prompt design). Although a formal evaluation of this method is beyond the scope of this work, recent research suggests that this analysis achieves state-of-the-art performance in terms of quality, accuracy, and granularity [59, 274], making it suitable for this work. However, it is important to note that utilizing `GPT-4` has limitations (at least at the time of writing of this document): 1) the process takes considerably longer compared to the NLI approach (seconds versus milliseconds), and 2) scaling is challenging due to strict request-per-minute (RTM) limits imposed by OpenAI, which hampers efficient handling of a large volume of requests.

Page-level overview of options and criteria. As evidenced by our formative study, providing a page-level overview of the information space to users can greatly assist them in reading and sensemaking tasks. To facilitate this, Selenite consolidates paragraph-level metadata into the sidebar's options and criteria entries, with the entries that are present on the page highlighted (Figure 7.1c&d). This offers two key benefits. Firstly, it provides a comprehensive summary of all the available options and criteria specific to the current page, which allows users to quickly understand the focus of the page as well as judge its value against their personal interests. Secondly, Selenite enables structured and effortless navigation. By utilizing the "locate previous/next" but-

[15]The model can be accessed on-demand through a remote API service that we implemented.

tons (Figure 7.3), users can swiftly move between distinct parts of the page related to identified options and criteria. This feature saves users time and effort, as it eliminates the need for manual searching and filtering, which our formative study found to be the common practice.

It is worth noting that the combination of page and paragraph-level annotations effectively addresses a significant limitation initially highlighted by Crystalline (see Chapter 4, section 4.3.2.1) and further revealed in our formative study: the inability to recognize "latent/implicit criteria," where the same criterion can be expressed in various forms without being explicitly mentioned. For instance, it involves identifying the criterion of "price" from a statement like "I bought this mp3 player for almost nothing" [292].

7.3.2.3 [D3] Dynamically Suggesting Next Steps in Sensemaking

Traditional sensemaking systems have aimed to assist users in managing and arranging their past experiences, such as collecting and organizing information that they have encountered [145, 181, 204, 232, 234, 235]. However, we propose that there is an untapped opportunity to utilize these activities to guide users' attention towards their next steps in sensemaking. Selenite offers two such "forward-looking" benefits. Firstly, it reminds users of potentially overlooked criteria once they finish reading a page. Secondly, it suggests exploring content related to criteria that users have not encountered or have limited evidence about, aiming to maximize information gain.

To achieve this second objective, we leverage users' reading activities in terms of the subset of criteria they **cared about** (determined by if a user did spend time dwelling on a particular paragraph [80], similar to the implicit behavior tracking that Crystalline [234] employs) and the subset they have intentionally **ignored** (those that exist on the page but users chose to skip reading about, i.e., did not spend time on)[16] to recommend additional *relevant* and *diverse* criteria to search for and read about from the remaining global list. This requirement for the suggested criteria to be both relevant and diverse is similar to the exploration-exploitation trade-off in information retrieval [36], and has also been extensively documented in recommender system literature — they help maintain user engagement and interest while avoiding over-fitting and filter bubbles [202, 336].

To operationalize this idea, we can consider it as a graph problem: By constructing a fully-connected graph using the global list of criteria as *vertices*, we assign *edge* weights as distances between respective criteria in a semantic embedding space and vertex weights as the criterion's

[16]The threshold of determining if the user indeed paid attention to a paragraph is set to 2 seconds based on our empirical testing. Future work can investigate more adaptive methods, such as taking into account the length of a paragraph, the amount of new information contained in a paragraph compared to users' existing knowledge, or if users appear to be idling and performing irrelevant activities.

relevance to the subset of criteria that the user cared about. Our objective then is to recommend a *diverse* subset of criteria (vertices) that have large distances between each other while still being *relevant* to what users cared about. That is, we need to find a sub-graph G' of size k, which maximizes a weighted ($\beta > 0$) sum of vertex weights w_V (relevance) and edge weights w_E (diversity):

$$\arg \max_{G' \subset G, |G'|=k} \beta \cdot w_V(G') + w_E(G')$$

To build the graph, we measure relevance with the perplexity score of the sentence "`{global_criterion} tend to be considered together (or is a trade-off) with {cared_about_criterion}`" using `GPT-2` [294][17], and characterize diversity with the cosine distance between the SentenceBERT [296] embeddings of the two vertices (criteria). Here, we follow the classic greedy peeling algorithm [368] by dropping vertices with the lowest weights (the sum of vertex and every edge weight) one at a time in a greedy fashion until the graph size reaches $k = 2$.

We provide suggested search keywords based on specific criteria (Figure 7.1f), allowing users to refine their queries (e.g., "comparison of JavaScript frameworks, `on community support`", or "best baby strollers, `on ease of assembly`"). This aims to help users find and review targeted information more effectively, ensuring a consistent and continuous acquisition of knowledge.

7.3.3 Implementation Notes

The Selenite browser extension is implemented in HTML, TypeScript, and CSS and uses the React JavaScript library [109] for building UI components. It is built on the Skeema system, similar to Wigglite. It uses Google Firebase for backend functions, database, and user authentication.

As mentioned before, we leverage `GPT-4` for several use cases. Due to length limitations, we document the specific prompt designs for those tasks in Appendix A, and only discuss a few challenges we experienced while interacting with the `GPT-4` API here in this section. First, due to the limited context window size of `GPT-4` (8192 tokens or approximately 6100 English words), we occasionally need to divide the entire text content of a web page into smaller chunks and run parallel queries to extract options. Unfortunately, as of the time of writing, we do not yet have access to the `gpt-4-32k` version of `GPT-4`. This version would theoretically accommodate ap-

[17] Perplexity is a measurement of how well a probability model predicts a sample [56]. In the context of natural language processing, it measures how well a language model predicts a given sequence of words. Here, we use perplexity to characterize how much a language model (e.g., `GPT-2`) is "surprised" by seeing a given sentence. If the perplexity score for the sentence is low, it means that the model can predict the sequence of words well based on its learned language patterns. In other words, the sentence is statistically likely and coherent according to the language model. Unlike `GPT-3.5` or `GPT-4`, who are only available in the form of an auto-completion API at the time of writing of this thesis, `GPT-2` is open-source, and we can therefore obtain the perplexity of a sentence directly as the exponent of its inference loss [170].

proximately four times more content, significantly reducing the need for chunking and parallel queries. Second, unfortunately, there are occasions when the `GPT-4` model becomes overloaded with requests or takes an exceptionally long time to respond. To mitigate these problems and provide uninterrupted user experience to Selenite users, we have employed the following two approaches: 1) *Dual API requests*: We send two identical requests using separate API keys simultaneously. We prioritize the response that returns first with valid information, indicating that it is not an error and contains the requested information from the prompt; 2) *Graceful error handling & retry*: In the event of an error, we introduce a random delay (ranging from 1 to 5 seconds) before retrying the request. We repeat this retry process for up to 5 attempts, allowing sufficient opportunity for a successful response. Note that these issues are attributable, in part, to the current limited beta status of `GPT-4`. Consequently, it is uncertain whether these issues will persist in the future. Nevertheless, we delve into them here to provide a comprehensive and accurate accounting of our experience interacting with the API.

To efficiently perform natural language inference (NLI) during the analysis of article content to produce per-paragraph summaries and annotations of options and criteria, we experimented with both the `roberta-large-mnli` and `bart-large-mnli` models that are fine-tuned for multi-genre natural language inference (MNLI) tasks[18], and ended up using `bart-large-mnli` due to its better performance in our informal testing. In addition, we implemented a REST API service that the Chrome extension can query on demand. To decrease model inference time and ensure a smooth user experience, we ran the service on multiple Google Cloud virtual machines with NVIDIA L4 GPUs, which proved to be at least 100x faster than inference using CPU alone, and can typically process the entire content of a single page (against around up to 10 different options and 20 different criteria) within 5 seconds.

As explained previously, we implemented Selenite using state-of-the-art NLP models: off-the-shelf `GPT-4` [274] and models finetuned on `BART` [214]. These models were chosen for their strong performance that would satisfy our prototyping needs as well as their generalizability across different application domains (c.f. Section 7.4). However, it is important to note that *our contributions lie more in the concept of grounded reading, interface design, and underlying NLP task abstractions, which are independent of specific model usage.* We anticipate that these designs will remain valid as AI techniques continue to advance [80].

[18]These two models are considered to be able to achieve state-of-the-art performance as of June 2023.

7.4 Study 1: Performance Evaluation

While Selenite can help ground users in what to read, its impact may backfire if the list of options and criteria is not accurate or comprehensive — Anchoring bias [371] may cause readers to more easily miss information that is *indeed included in the page* but *not* reflected in the generated options and criteria list. Here we evaluate whether Selenite can:

1. *accurately* report options that are present on a web page,
2. *comprehensively* report critical criteria people commonly consider,

by testing it on a set of diverse topics.

7.4.1 Methodology

7.4.1.1 Topic Sampling

We collected ten topics that exhibit a mixture of practicality and diversity (Table 7.3): (1) we randomly sampled 5 topics (out of the 28 topics reported) reported by participants in the formative study, and (2) we collected 5 more from Wirecutter, a popular review site — the three most popular product guides listed in their 2021 year-in-review (at the time of writing, the 2022 year-in-review has not been published) as well as their two most recently updated guides for June 2023.

7.4.1.2 Groundtruth Dataset Creation for Options and Criteria

To collect groundtruth criteria that the general audience would care about for each topic, the first author mimics a typical information collection workflow, where people rely on top sources from popular search engines for their authenticity and credibility. Specifically, we first gathered the top five Google search results using the query template "`best [product or category]`" (excluding promotions or ads). Then, for each web page, we took a two-pass approach, where we first read through and annotated the options and criteria mentioned in every paragraph, and then merged all the annotations, excluding duplicate ones. Note that since many criteria are mentioned in a descriptive manner (e.g., the phrase "It is available in a black finish" implicitly refers to "aesthetics"), we had some variance in how we named essentially the same criteria in the first pass. We manually merged the similar ones if we believed semantically they refer to the same concept.[19] Finally, for the topics that we sampled from the formative studies where participants explicitly collected options and criteria using Unakite, we double-checked and were able to verify

[19]We made our best effort to maintain objectivity throughout this task. However, we faced time constraints that prevented us from obtaining an independent and reliable second opinion, which we will pursue shortly after the submission of this thesis.

Topic	Count		Topic-Level			Paragraph-Level		
	#Groundtruth	#Selenite	Precision	Recall	F1	Precision	Recall	F1
Best washing machines	19	24	0.88	1.0	0.93	0.91	1.0	0.95
Birthday gift ideas	11	21	0.57	0.91	0.70	0.57	0.96	0.72
Best hybrid app frameworks	15	21	0.86	0.93	0.89	0.83	1.0	0.91
Best time tracking tools	21	21	0.81	0.95	0.88	0.88	0.98	0.93
Deep learning frameworks	25	20	0.80	0.84	0.82	0.87	0.95	0.91
Best sleeping bags	19	21	0.81	0.89	0.85	0.95	1.0	0.97
Best air purifiers	20	24	0.83	1.0	0.91	0.83	0.98	0.90
Best robot vacuums	23	28	0.82	1.0	0.90	0.95	1.0	0.97
Best baby strollers	22	24	0.92	1.0	0.96	0.81	1.0	0.90
Best tropical vacation spots	15	19	0.74	0.93	0.82	0.92	1.0	0.96
Mean	19.0	22.3	0.80	0.95	0.87	0.85	0.98	0.91

Table 7.3: Results for the performance evaluation on Selenite's capability to retrieve a high-quality set of commonly considered criteria by topic.

that all the criteria that participants identified were indeed included in our groundtruth dataset, providing preliminary evidence to the soundness of our groundtruth dataset.

7.4.1.3 Evaluation Metrics

Option Extraction Since in Selenite, we directly extract options from web pages, we evaluated this capability using the *accuracy*, that is, the percentage of options extracted by Selenite out of all the options available on a page.

Criteria Retrieval We also evaluated Selenite's ability to retrieve the right set of criteria on two levels. First, to answer whether Selenite helps find useful criteria *for each topic*, we compute *topic-level precision* ("the fraction of criteria retrieved by Selenite that were in the groundtruth") and *recall* ("the fraction of groundtruth criteria that are were retrieved by Selenite").

Second, to measure whether Selenite provides high-quality groundings *per paragraph*, we additionally randomly sampled 20 paragraphs per topic, and computed *paragraph-level precision* ("the fraction of criteria recognized by Selenite that were indeed mentioned in the paragraph") and *recall* ("the fraction of criteria mentioned in the paragraph that were recognized by Selenite").

7.4.2 Results

7.4.2.1 Option Extraction

Selenite achieved 100% accuracy on extracting options from web pages, i.e., as long as there was an option explicitly mentioned on a web page, Selenite was able to correctly extract it. This directly

speaks to the strong reasoning and information extraction capabilities of `GPT-4` as described in OpenAI's technical report [274].

7.4.2.2 Criteria Retrieval

We present the result of criteria retrieval evaluation metrics in Table 7.3, which provides initial evidence to Selenite's strong capability in presenting to the user a comprehensive set of criteria that people commonly consider. Notice that for most of the topics, Selenite was able to retrieve more criteria compared to the groundtruth set. This is not surprising, partly due to the fact that `GPT-4` has likely synthesized information from significantly more sources than what was considered during the construction of the groundtruth dataset (five web pages for each topic). Theoretically, there is also a possibility that `GPT-4` hallucinated some criteria that are largely irrelevant to a given topic, however, upon further manual inspection, we did not see evidence of hallucination, at least for the 10 topics considered in this performance evaluation (for example, Table 7.2 shows a list of commonly considered criteria that Selenite retrieves for the topic of "best baby strollers").

Topic-level recommendations Selenite achieved both high recall and high precision on multiple topics (e.g., *best washing machines*, *best baby strollers*, and *best robot vacuums*), and usually achieves higher recall than precision, suggesting that Selenite has the tendency of finding *supersets* of what users would generally be able to identify from reading, i.e., criteria in the groundtruth set.

We qualitatively analyzed the topics with a lower-than-average topic-level criteria recall, and found two contributing reasons: (1) Some web pages cover factual information that is not necessarily relevant. Multiple pages describing *Best Hybrid App Frameworks* mentioned *First Release Date*, which arguably is not a *criterion* necessary for selection. (2) Some criteria are intercorrelated. For example, in the case of *deep learning framework*, whereas it did not explicitly mention "`growth speed`," Selenite did suggest "`innovation`," whose description is "`the ability of the framework to stay up-to-date with the latest research and developments in deep learning, and to incorporate new techniques and architectures as they emerge.`" While we did not count these two as equivalent in the evaluation, in practice, these two have a high correlation, and we believe having one included might be sufficient. Still, this potential mismatch reflects the necessity of allowing users to edit the criteria and descriptions.

Meanwhile, upon initial observation, Selenite's lower precision on certain topics may suggest its inclination towards retrieving unnecessary criteria. However, a closer examination revealed

an interesting insight: For instance, when it comes to topics like *birthday gift ideas*, popular web pages often present a list of 10+ diverse options that lack strict comparability and are all described using generic terms such as "fun" or "sweet". This lack of specificity makes it challenging to determine a comprehensive set of groundtruth criteria. In contrast, Selenite offers comprehensive overviews that encompass factors like `personalization, uniqueness, practicality, sentimentality, and presentation (wrapping)`, among others.

Paragraph-level Grounding Selenite also achieved high per-paragraph performances, again with a bias towards higher recalls. This is intentional — we tuned the parameters of the NLI-based method such that it is more likely for Selenite to claim non-existing criteria than overlooking actual existing ones. This approach prioritizes avoiding information loss, which, suggested by prior work [236], is a more expensive mistake compared to user verification. We order the criteria based on their probability score from the NLI model and will, in future iterations, fade the ones with a lower score.

We did notice that in some rare circumstances, the NLI performance can be influenced by a criterion's description, e.g., changing "appropriate for age" to "appropriate for kids, adults, or elderly" can reduce Selenite's error on recognizing arbitrary numbers as ages. Therefore, in future iterations of Selenite, we will provide a hint to users, prompting them to try tweaking the description when they attempt to delete a criterion due to its seemingly low grounding efficacy.

7.5 Study 2: Usability Evaluation

In addition, we conducted an initial usability study to verify if the features provided by Selenite are usable and if the approach of providing global as well as contextual grounding offers can allow users to read, navigate, and comprehend information more efficiently. Specifically, we were interested in the following quantitative research questions:

- **[RQ1]** Does using Selenite speed up people's process of reading and understanding information?
- **[RQ2]** Does using Selenite help people achieve a more comprehensive understanding of an information space?
- **[RQ3]** Can Selenite help people obtain new information in addition to their existing knowledge?

7.5.1 Participants

We recruited 12 participants (5 female, 7 male) aged 21-40 (mean age = 28.9, SD = 5.2) through emails and social media. Participants were required to be 18 or older and fluent in English. All participants reported that they regularly engage in the process of seeking and sifting through large volumes of online information, whether for professional or personal purposes, on a weekly basis.

7.5.2 Procedure

The study was a within-subjects design, where participants were presented with two tasks and were asked to complete each one under a different condition, counterbalanced for order. For each task, participants were given a topic that they needed to investigate and two web pages relevant to the topic that they were required to read and process. The two topics were *"best baby strollers"*[20] and *"best robot vacuums"*[21]. The provided two web pages for each respective topic were all product comparison pages used in the performance evaluation (see section 7.4.1). For each task, participants were asked to read through the two required pages, either by themselves without any aid (a *control* condition simulating how people normally read) or with Selenite (*experimental* condition). While reading, they were instructed to write down *as many* criteria as they learned and thought were important for the topic as well as the reason why they were important as if they needed to thoroughly explain the topic to a friend later. After finishing the two pages, participants were instructed to optionally search (using Google) and gather additional information that they still wanted to learn about but weren't able to from reading the required pages. We imposed a 25-minute limit per task to keep participants from getting caught up in one of the tasks. However, they were instructed to inform the researcher that they felt like they could make no further progress, i.e., having learned as much as they could about the given topic.

To ensure realism and participant engagement, the tasks were selected based on actual topics that the formative study participants reported investigating. Rather than letting participants search for their own pages to read from the get-go, we provided them with a predefined set of pages to enable a fair comparison of the results (e.g., speed, etc.). Requiring participants to use predefined pages (each contains, on average, 15 screenfuls of content) for the first portion of the study also helps ensure that the two tasks are of roughly equal difficulty in terms of reading and

[20]The two web pages that participants were required to read for "best baby strollers" are: 10 Best Strollers of 2023 | Tested by GearLab and 11 Best Baby Strollers of 2023, Tested by Parents & Experts

[21]The two web pages that participants were required to read for "best robot vacuums" are: The best robot vacuum cleaners to get in 2023 - The Verge and Best robot vacuums in 2023 tested and rated | Tom's Guide

cognitive processing effort. As described in the results, there was no significant difference by task.

Each study session started by obtaining consent and having participants fill out a demographic survey. Participants were then given a 5-minute tutorial showcasing the various features of Selenite and a 5-minute practice session before starting. At the end of the study, the researcher conducted a NASA TLX survey and a questionnaire, eliciting subjective feedback on their experience in both conditions. Each study session took approximately 60 minutes, using a designated Macbook Pro computer with the latest version of Chrome and Selenite installed, and was conducted remotely via the Zoom video conferencing software. The study was approved by our institution's IRB office.

7.5.3 Results

All participants were able to complete all of the tasks in both conditions, and nobody went over the pre-imposed time limit. Below, we compile together both quantitative and/or qualitative evidence to evaluate Selenite with respect to our research questions.

First, we were interested in understanding if Selenite can help participants read and process information faster compared to the baseline condition (RQ1). To examine this, we measured the time it took for them to finish reading all the materials in each task. A two-way repeated measures ANOVA was conducted to examine the within-subject effects of the condition (baseline vs. Selenite) and task on completion time. There was a statistically significant effect of condition ($F(1, 20) = 102.5$, $p < 0.01$) such that participants completed tasks significantly faster (36.3%) with Selenite (Mean = 840.3 seconds, SD = 102.7 seconds) than in the baseline condition (Mean = 1319.3 seconds, SD = 120.0 seconds). There was no significant effect of task ($F(1, 20) = 0.40$, $p = 0.53$), indicating the two tasks were indeed of roughly equal difficulty. These results suggest that Selenite helped participants read and comprehend information more efficiently. We provide additional qualitative insights into why Selenite was more efficient in the following open-ended user study (section 7.6.3).

In addition, we were interested in understanding if Selenite can help participants achieve a more comprehensive understanding of a topic (RQ2). To measure this, we first compared the *quantity* of criteria that participants externalized under each condition. As a pre-filtering step, we rated all the criteria that participants externalized as either *valid* or *invalid* blind to the conditions. Valid criteria are considered as ones that are *relevant* to the topic and *backed by specific evidence* that can be traced back to the content, consistent with those standards used by prior work in judging the quality of subjective evidence [73]. After filtering out the criteria that were

	Mental demand	Physical demand	Temporal demand	Performance	Effort	Frustration
Selenite	3.0 (3.03 ± 1.76)*	1.0 (0.51 ± 1.74)	2.5 (2.26 ± 1.68)*	8.5 (8.47 ± 1.32)*	3.5 (4.08 ± 1.88)*	0.5 (0.33 ± 1.51)
Baseline	6.5 (6.43 ± 2.07)*	1.0 (0.79 ± 1.98)	4.0 (4.29 ± 2.08)*	6.5 (6.54 ± 1.73)*	6.0 (5.98 ± 2.23)*	1.0 (0.89 ± 1.91)

Table 7.4: Study 2 participants' responses to NASA TLX questions (on a scale from 0 to 10) in study 2. Format: median (mean ± standard deviation). Statistically significant differences ($p < 0.05$) through t-tests are marked with an *.

invalid, we found that the average total number of valid criteria increased by 90.4% when using Selenite (Mean = 12.93, SD = 3.90) compared to the baseline condition (Mean = 6.79, SD = 4.07), which is statistically significant ($p < 0.01$) under a t-test. Thus, using Selenite appeared to enable participants to identify and learn significantly more criteria about a topic compared to people's current way of reading information.

In addition to quantity, we also examined the *quality* of the criteria by comparing the ones that participants externalized with the groundtruth criteria curated in the previous performance evaluation — we can calculate the precision (calculated as $n_{\textbf{Hit}}/n_{\textbf{Total}}$) and recall (calculated as $n_{\textbf{Hit}}/n_{\textbf{Groundtruth}}$) of participants' criteria that *hit* the groundtruth (where $n_{\textbf{Total}}$ is the total number of valid criteria participants externalized, and $n_{\textbf{Groundtruth}}$ is the number of groundtruth criteria for each task). On average, participants in the Selenite condition achieved significantly ($p < 0.05$) higher precision (98.8% vs. 78.4%) as well as significantly ($p < 0.05$) higher recall (73.0% vs. 30.4%) in both tasks. Thus, using Selenite appeared to have enabled participants to improve the quality of their understanding of an information space in terms of its criteria.

Furthermore, to understand if Selenite can help participants obtain new information in addition to what they have already learned from reading the two required pages (RQ3), we examined: 1) the number of additional searches that they performed in the Selenite condition (Mean = 2.01, SD = 1.39), which turned out to be significantly more ($p < 0.05$) than the baseline condition (Mean = 0.33, SD = 0.62); 2) the number of additional pages visited in the Selenite condition (Mean = 2.76, SD = 2.32), which turned out to be significantly more ($p < 0.05$) than the baseline condition (Mean = 0.42, SD = 0.74); and 3) the number of additional criteria that participants externalized in the Selenite condition (Mean = 1.58, SD = 0.91), which turned out to be significantly more ($p < 0.05$) than the baseline condition (Mean = 0.33, SD = 0.22). These results suggest that Selenite did encourage and help participants to seek additional information beyond their existing perspective.

Last but not least, participants filled out a NASA TLX [148] cognitive load scale and a System Usability Scale (SUS) [213] questionnaire for each condition. SUS Likert items were integer-coded on a scale from 1 (strongly disagree) to 7 (strongly agree). The median response values are presented in Tables 7.4 and 7.5. Notably, participants perceived Selenite to have significantly lowered workload across mental, temporal, and effort demands as well as significantly increased perceived

Question category	Statement	Response
Comprehensibility	I would consider my interactions with the tool to be understandable and clear.	6 (6.33 ± 1.10)
Learnability	I would consider it easy for me to learn how to use this tool.	7 (6.71 ± 1.04)
Enjoyability	I enjoyed the features provided by the tool.	6 (6.13 ± 1.72)
Applicability	Using this tool would make solving sensemaking problems more efficient and effective.	6 (6.28 ± 1.39)
Recommendability	If possible, I would recommend the tool to my friends and colleagues.	6 (6.23 ± 0.94)

Table 7.5: Study 2 participants' responses to System Usability Scale questions (on a scale of 1 to 7, where 1 represents "strongly disagree" and 7 represents "strongly agree") in study 2 regarding their Selenite experience. Format: median (mean $\pm$ standard deviation)

performance based on paired t-tests). This suggests that using Selenite can reduce the cognitive load and interaction costs when reading and understanding information, even when users had to learn and get used to a new user interface.

7.6 Study 3: Open-ended Case Study

Furthermore, we conducted an open-ended case study to understand the usefulness and effectiveness of the Selenite prototype from a qualitative perspective.

7.6.1 Participants

We recruited 8 participants (3 male, 5 female; 3 students, 2 software engineers, 1 medical professional, 1 accountant, and 1 researcher) aged 24-55 years old (mean age = 33.6, SD = 8.1) through emails and social media. Participants who participated in the previous usability study were excluded from this study. Participants were required to be 18 or older and fluent in English. All participants reported that they regularly engage in the process of seeking and sifting through large volumes of online information, whether for professional or personal purposes, on a weekly basis.

7.6.2 Methodology

Each participant first completed two pre-defined tasks, where they were instructed to use Selenite to help them read information about an unfamiliar topic. From the total 28 topics that participants reported having explored in the formative study (see Table 7.1), we randomly selected two that the participant was unfamiliar with (indicated in their screening survey). For each task, participants were presented with a set of three web pages that cover the topic that the formative study participants had gone through respectively. The provided web pages were primarily review arti-

Participant	Topics	Participant	Topics
P1	Choosing a high-quality digital camera Choosing the best air purifier Picking a suitable hand truck	P5	Picking the best robot vacuum Picking a drone Choosing the best e-reader
P2	Selecting the best washing machine Choosing a reliable VPN provider Deciding on unique thank-you gifts	P6	Picking the right refrigerator Choosing the best barbecue grill Choosing the best tropical vacation location
P3	Selecting the best mattress Choosing the best city bike Choosing a birthday gift	P7	Choosing an effective diet plan Choosing a breed of dog to adopt Selecting a suitable SUV
P4	Selecting a secure password manager Choosing the best tropical vacation location Choosing a hybrid app framework	P8	Picking a reliable treadmill Picking the right wedding venue Choosing the best skiing venue

Table 7.6: Topics that participants in study 3 explored.

cles comparing several options together or product detail pages. We imposed a 15-minute limit per task to keep participants from getting caught up in one of the tasks.

To further explore Selenite's potential, all participants were then instructed to use Selenite to help them make sense of a third topic that they intend to explore in real-life. In order to encourage the exploration of potential new insights, we purposefully did not limit the topic to be unfamiliar to the participants. This allows participants to optionally revisit previous topics of interest and potentially uncover fresh perspectives.

Each study session began by obtaining consent and having participants fill out a demographic survey. Participants were then given a 5-minute tutorial showcasing the various features of Selenite and a 5-minute practice session before starting. At the end of the study, the researcher conducted a semi-structured interview, eliciting subjective feedback on the experience of using Selenite. The interviews were recorded and transcribed, after which qualitative coding and thematic analysis [77] were performed. Each study session took approximately 60 minutes, using a designated Macbook Pro computer with the latest version of Chrome and Selenite installed, and was conducted remotely via the Zoom video conferencing software. The study was approved by our institution's IRB office.

7.6.3 Results

Table 7.6 documents the topics that participants explored during the case study.

Below, we present the major qualitative findings from the observation of participants' behaviors using Selenite as well as their feedback from the post-study interviews.

Time and effort savings. All of the participants mentioned that using Selenite would save them a lot of time and effort compared to using their typical reading and information collection workflow, echoing the quantitative results reported in the usability evaluation (see section 7.5). First of all, having access to the global overview felt like "*a game-changer*" (P8) that offers a "*bird's-eye view*" (P4) or access to "*on-demand expert opinion*" (P1) that "*took away the anxiety and guesswork of wondering what other folks would actually care about*" (P5). P7 suggested that "*this is something that I always wished for when reading about stuff that I'm not an expert in. It seriously saves me a ton of time that I'd otherwise spend trying to wrap my head around it little by little,*" while P6, who couldn't "*stand the whole deal of figuring out stuff that I'm not used to*" said "*now I really feel like I'm chilling in the passenger seat and not having to do all the heavy-lifting stuff personally.*"

Second, participants seemed to appreciate the in-context annotations and summaries of each paragraph provided by Selenite. They thought that this feature "*made things incredibly easy*" (P3) by "*helping me grasp the key points without wasting time reading a paragraph through*" (P1), and "*felt like back in the day when my classmate would mark all the important stuff in the textbook after a class when I couldn't make it.*" However, some did report that the in-context annotations can occasionally be "*a little bit distracting*", especially for paragraphs that are "*apparently unrelated to the main content*" (P2), such as those that talk about related articles or terms of services, suggesting that future versions of Selenite should consider more robust content filtering techniques.

Last but not least, participants also appreciated the fact that Selenite can help them brainstorm search queries that would enable them to find new information more efficiently that was "*almost always one step ahead*" (P4), especially in the third task. For example, after reading two review articles about e-readers, Selenite suggested that P5 could do some additional investigations about the "supported file formats" and "syncing across devices." P5 admitted that "*I'd totally miss those things if I'm by myself, and even if I'm trying to be super careful, it would take me forever to figure out that I need to check out those aspects.*" Additionally, we observed that when integrating the Selenite suggested criteria into subsequent search queries, the search engine did return result pages that turned out to be noticeably different yet sufficiently high-quality for users to explore.

Impact on reading patterns and habits. Participants all mentioned that they immediately checked out the commonly considered criteria from the sidebar before diving into reading the first web page. They claimed that compared to what they normally do, which is "*just have to hunker down and read*", reading the overview first instead helped them "*cut to the chase and get a feel of what's out there*" (P7) and remind them of criteria that would otherwise "*slip [their] mind*" (P3).

On a per-paragraph level, we noticed an initial hesitation among some participants (3 out of 8) towards relying solely on the provided criteria labels. As a safety precaution, they personally read through a handful of paragraphs to confirm the labels' accuracy and reliability. We further corroborated this observation with their reflections, such as "*I've never seen anything like this before, so honestly, I was a bit skeptical at first. But hey, everything looked legit!*" (P5) After this initial hurdle, participants tend to "*rely on the labels to tell me the gist of a paragraph*" (P4) and only read paragraphs that discuss criteria that they truly cared about. For the content that participants did end up reading, they think the corresponding criteria "*definitely helped me process and digest it better*" (P6), and even "*saved me from otherwise misunderstanding things*" (P7). For example, while exploring healthy diet plans, P7 reflected that he would have initially thought a paragraph detailing the caloric allocation for each meal was seemingly discussing "calorie intake," however, Selenite preemptively clarified that the focus was on "portion control," i.e., "providing guidelines on portion sizes."

Participants also enjoyed the easy navigation feature that Selenite offers, and used it to frequently jump between different criteria mentions for easy comparison and digestion (7/8). They claimed that finding specific criteria about different options in a long article used to be "*finding a needle in a haystack*" (P8) that they were hesitant to do, but with Selenite, "*it's more like following a well-lit path*" (P5). For example, P2 reflected on her experience exploring VPN solutions, and claimed that "*now I get it, McAfee Safe Connect seems to be keeping track of all sorts of my information while SurfShark doesn't do any of that. If I can't quickly switch between these two points on the page, by the time I reach SurfShark's no-logging policy, I would have totally forgotten about what McAfee does, or that I should even be concerned about no-logging at all.*" In addition, participants liked the fact that they can more effectively break out from the original structure and narrative of an article; for instance, P1 recounted that "*you don't gotta stick to what the authors say anymore, ya know? Because, let's face it, their storylines can get all tangled and complicated sometimes.*"

Last but not least, we did not observe much usage of the "zoom in" feature, where Selenite can leverage `GPT-4` to provide a thorough analysis of a piece of content — only 3 participants tried it for a total of 8 times. We hypothesize that 1) the web pages utilized in the study were all professionally crafted, resulting in content that was relatively easy to comprehend; 2) the criteria labels generated by our NLI pipeline proved to be adequate in addressing the participants' information needs; 3) the time required for the "zoom in" feature to provide a useful analysis, typically ranging from 5 to 10 seconds, still exceeded the participants' patience and attention span. Future work could explore solutions to address this limited adoption from these perspectives.

Additional findings. One interesting theme that emerged was that participants (4/8) opted to use Selenite in the third task to revisit topics that they had previously explored before and wanted to be able to "*double-check*" (P3) whether their prior understanding of the topic was truly comprehensive. Consistently, each participant uncovered something new that they hadn't considered before. For example, P3, who had recently been making plans to move in with his partner, revisited the topic of "choosing the right mattress," and realized that he had never taken into consideration criteria such as "motion transfer" (i.e., the extent to which movement on one side of the mattress affects the other side) or "noise reduction" (i.e., the ability of the mattress to minimize noise from springs, coils, or other components), which prompted him to reassess his original mattress purchase. As another example, P4, a professional software engineer, revisited the topic of "choosing a hybrid app framework" and discovered that he had neglected to consider the "Licensing and legal considerations" (i.e., compliance with licensing requirements and legal considerations) as suggested by Selenite. Consequently, P4 was able to find additional evidence to confirm the validity of their original framework choice made back in 2017.

In the post-study interview, many participants (6/8) felt that now they "*can't imagine reading without a tool like this (Selenite)*" (P3). Half even inquired about the possibility of installing Selenite on their personal computers for post-study usage, and we gladly fulfilled their requests. Despite encountering a few bugs in our research prototype during the study and having no obligation or incentives for continued usage after the study, the fact that they were willing to do so suggests that our grounded reading approach indeed holds value for our participants.

7.7 Discussion

Some of the participants (3/8) from the open-ended user study expressed concern about the coverage of Selenite's overview criteria and the criteria labels for each paragraph. They wondered if Selenite might overlook important criteria that they should also consider. This concern was valid, given that we presented the tool as an AI-powered oracle that could potentially be fallible or overlook certain factors and encouraged users to conduct their own explorations in addition to relying on Selenite's insights. Our performance evaluation described in section 7.4 provides initial validation that the criteria and options provided by Selenite are indeed comprehensive, relevant, and accurate. However, participants also acknowledged that the current set of criteria offered by Selenite already "*far exceeds what they could identify and keep track of on their own*" (P4); therefore, they "*wouldn't mind at all if the algorithm misses any minor ones*" (P1). Indeed, despite the participants' awareness of the opportunity to request additional criteria from Selenite in

the case of insufficient coverage (as confirmed in the post-study interviews), we did not observe any instances of such usage. Nevertheless, further research is necessary to unlock: 1) methods that would further enhance the coverage and accuracy of Selenite, such as leveraging retrieval-augmented models; 2) mechanisms and interventions designed to reduce over-dependence on Selenite as well as encourage user-led explorations with critical thinking.

Though framed as a tool for grounded reading, Selenite may also have the potential to address some of the issues identified by prior work as well as in previous chapters regarding structuring information — prior research suggested that asking users to structure information too early might lead to a more poorly structured information space [193]. In addition, the knowledge structures that people created often become obsolete, and new structures often emerge as their mental representations evolve over the course of their investigation [114, 152, 193], resulting in having to spend significant effort in refactoring the structures. Here, Selenite provides users with a well-structured framework from the outset, including a set of commonly acknowledged criteria. This readily usable scaffold serves as a starting point, aiming to encompass the majority's perspective and thereby minimizing the necessity of refactoring or restructuring. Hopefully, it simplifies the iterative and cognitive-demanding process of building a mental model, transforming it into a possibly more manageable task of refining and pruning [234].

There was also a concern that `GPT-4` could potentially hallucinate or generate irrelevant or even false criteria and thus mislead users in their subsequent exploration. However, it is important to note that in our case study (as well as in study 2), we did not observe such episodes or evidence of this occurring. This could be attributed to the fact that the topics explored in the study were all common subjects with abundant source materials available online, which were likely encountered by `GPT-4` during its training process. We would like to further conjecture that even if hallucination occurs, users can readily identify irrelevant or false criteria by carefully reading their descriptions and comparing them with common sense or their intuitive knowledge about the topic, mitigating the actual impact of hallucination.

7.8 Limitations & Future Work

Availability of domain knowledge in LLMs. LLMs, such as `GPT-4`, possess an extensive range of encoded knowledge, yet they might lack domain-specific information for specialized or newly emerging topics, as well as for topics involving confidential or sensitive information. Our technical implementation in Selenite is primarily based on extracting knowledge from LLMs (e.g., commonly considered criteria), and its effectiveness is highly dependent on the LLM's capability

to capture relevant domain knowledge from its training data. Without such knowledge, the guidance provided may be subpar. In addition, LLMs themselves can sometimes be biased, and the response it generates might be incorrect or harmful [201, 267]. However, our approach to ground the reading process with domain knowledge would also work with other sources of knowledge bases as well, for example, Unakite + Strata tables [232, 233], or crowdsourced [144, 252], or a combination of them. Furthermore, we should also urge users to thoroughly examine the Selenite overview when dealing with critical situations.

Correlations and Hierarchies among Criteria In Selenite, we made the implicit assumption that criteria and options are completely independent. However, this assumption doesn't always hold true. On the one hand, for instance, when evaluating the "Best Baby Stroller," the specific criterion of "suspension system" falls under the broader category of "safety," while on the other hand, aspects like "price" and "versatility" are typically trade-offs that are impractical to optimize for simultaneously.

Currently, Selenite takes into account one form of criteria correlations, i.e., relevance between criteria, when suggesting the next steps. This proved promising in the study, which gave us reasons to believe that further exploiting these connections between criteria can better support users' reading. For instance, instead of presenting the criteria in a simple list, we can create a behind-the-scene knowledge graph where criteria are connected using edges of relations (`TypeOf`, `CompetesWith`, etc.). By initially displaying a portion of this graph and allowing users to zoom in on the specific criteria they're interested in (e.g., a subset of "safety"-related features), we can help users intuitively reason through an initially overwhelming list. In addition, one can again imagine "overview first, details later"-style UIs [323] that incorporate criteria hierarchies, e.g., multi-level tables or lists, granting users the flexibility to combine or decompose criteria at decision time.

Study limitations. Considering the limited training and practice time, participants in our studies may not have had sufficient opportunity to become fully familiarized with the features offered by Selenite. Some of the study tasks and topics might not be the ones that participants typically encounter, and therefore they may not have sufficient motivation or background context as in real life. We partially addressed these by 1) choosing the training and real study task topics from the ones that formative study participants reported dealing with in real-life; 2) having participants use Selenite for their own work or personal tasks and projects at the end, which would presumably fuel them with the necessary motivation and context and engage with Selenite in a more organic way. There was also a risk of participants already being familiar with the topic they investigated in the study, such as an expert photographer choosing a digital camera. However,

in the post-study interviews, we confirmed that none reported having extensive experience or expertise in any of the task topics that they were assigned to explore.

Impact on learning. The current design of Selenite functions as an "index" to direct users to relevant parts of a web page for reading and processing. However, we need to be cautious about a potential risk associated with this approach — some users might believe they have gained sufficient knowledge about a topic by merely reading the overview and may, therefore, skip engaging with the actual web content. This behavior could lead to incomplete, biased, or even inaccurate understandings of the subject. It's akin to only reading the table of contents or indices of a book without delving into the actual content. Nevertheless, our studies conducted under controlled settings have shown that participants did, in fact, engage with the actual web content after going through the overviews. To build on this promising evidence, future research should additionally investigate interface and interaction design strategies that motivate users to explore and read the actual web content with the assistance of Selenite-style guidance. One potential approach could be progressively revealing criteria information to users based on their reading behavior, encouraging deeper exploration and understanding (discussed in detail in Chapter 8).

Chapter 8

Conclusion & Future Work

Here, I first summarize the contributions made throughout this dissertation, then reflect on the important lessons learned. Finally, I discuss a series of opportunities for future work to explore.

8.1 Summary of Contributions

The series of work introduced in this thesis points to the importance of having tool support that helps users efficiently and effectively read, comprehend, collect, and organize information to make and justify decisions, while automatically capturing the sensemaking context to help subsequent users understand and evaluate those decisions.
Specifically, my work made the following contributions:

- A thorough review of the background and related research on sensemaking in general and the various tools and systems for foraging, structuring, evaluating, and reusing knowledge (**chapter 2**).
- Unakite, a prototype system that: 1) reduces the costs of capturing and organizing online information in-situ, and 2) preserves the knowledge as design rationale to help subsequent users understand [166, 232] (**chapter 3**).
- Evidence that it is possible to automatically identify options, criteria, and relevant evidence from web pages that a user is browsing using a set of natural language understanding heuristics [234] (**chapter 4**).
- A set of implicit behavioral signals that users exhibit when browsing the web which can be leveraged for prioritizing and filtering that collected information [234] (**chapter 4**).

- A prototype system called Crystalline that integrates the heuristics and behavioral signals to automatically collect and organize viewed information into list and comparison table views for subsequent decision making [234] (**chapter 4**).
- A novel class of wiggle-based gestures that are cognitively and physically lightweight to perform to collect information, and can simultaneously encode aspects of users' mental context [235] (**chapter 5**).
- A prototype event-driven JavaScript library that implements such wiggle-based gestures and runs in web browsers [235] (**chapter 5**).
- Wigglite, a prototype system that takes advantage of the wiggle-based gestures to enable information capturing and classification during sensemaking that works on both desktop and mobile devices [235] (**chapter 5**).
- A synthesized framework for augmenting judgments of appropriate knowledge reuse including three major facets: context, trustworthiness, and thoroughness, as well as a prototype system called Strata that automatically records, computes, and visualizes many of the appropriateness signals described in the framework [233] (**chapter 6**).
- Evidence from a formative study showing people's need for support in terms of finding and reading information despite recent advances in information collection and organization tools (**chapter 7**).
- Selenite, a novel system providing users with a comprehensive overview of the information space upfront to jumpstart as well as guide their subsequent reading and sensemaking processes (**chapter 7**).
- A series of performance evaluation and user evaluations showcasing the feasibility, usability, usefulness, and effectiveness of our tools in reducing costs and increasing the benefits of externalizing people's mental models when sensemaking (**chapter 7**).

8.2 Discussion & Take-aways

Through designing, building, and evaluating the five systems described in this thesis, we gained deeper insights into the cost and benefit structure of sensemaking online. Below, I briefly reflect on the important lessons learned from this thesis.

8.2.1 Minimizing the interaction overhead

From the outset, I identified through formative studies that there's a high interaction overhead cost associated with people's current sensemaking processes — not only do they need to review various sources to find, collect, and organize information about different options and criteria while searching and browsing, which by itself is a cognitively and physically demanding task, but also the tools that they use nowadays poorly support the constant shifts between collecting, extracting, organizing, and reorganizing that are required.

Therefore, Unakite, Crystalline, and Wigglite all explored aspects of this concept of lowering the interaction cost. Unakite makes a first attempt at this by bringing the collection and organization of information from out-of-context note-taking apps to an in-situ sidebar, effectively *eliminating the need for context switches.* Crystalline took an additional step forward by leveraging the behavioral patterns that people naturally exhibit when reading and browsing to implicitly keep track of information on behalf of the user. This represents an attempt to *minimize away the interaction overhead using automation.* In the meantime, I also tried to push toward the other end of the "automation v.s. direct manipulation" spectrum by asking the question of "How can we create a seamless and fluid interaction that not only avoids interrupting users' reading flow (a common issue with existing tools) but also provides the necessary flexibility and expressiveness to capture users' judgments while gathering relevant content?" One viable answer is the wiggle gesture introduced by Wigglite, which can indicate the information to be collected and its valence, using a single, lightweight gesture that does not interfere with other interactions that are already available. This, again, tries to minimize the interaction overhead by *combining otherwise multiple interactions into a single one.* These tools, together with their success in their respective user studies, point to an important direction for the design of sensemaking interaction: by minimizing the cognitive load and physical effort required from the user, we can *transform the cost structure of sensemaking and encourage users to externalize more of their mental models.*

8.2.2 Providing appropriate augmentation of context

Reflecting back, I would like to argue that *appropriate augmentation of context* can be useful both for situations where an individual first approaches an unfamiliar topic by themselves and scenarios where they have to evaluate someone else's sensemaking results for potential reuse. In the first case, I discovered through contextual inquiries and retrospective walkthroughs that people often struggle with identifying key aspects in a traditional "bottom-up" fashion when encountering unfamiliar topics. In addition, once they become familiar with the topic, they face difficulties

in finding additional information to broaden their perspectives. Here, I leverage comprehensive overviews synthesized by state-of-the-art large language models to provide users with a global grounding to jumpstart and augment their sensemaking processes (Selenite). In the second scenario, where a subsequent person faces challenges in evaluating the sensemaking outcomes of a previous person for potential reuse, I developed a set of techniques for capturing the initial person's behavior and visualizing signals calculated based on the behavior. Such visualizations are used to augment the subsequent person's reuse assessment, enabling them to effectively aggregate and build on existing sensemaking results.

8.2.3 Integrating top-down and bottom-up approaches

While the final system, Selenite, promotes a top-down grounding approach, it is important to consider the potential negative impacts of over-grounding. During our interface design and performance evaluation, we initially validated the feasibility of providing a comprehensive overview while acknowledging the potential uncertainties of the suggested results. However, one could argue that iteratively constructing mental models from scratch offers valuable learning and internalization opportunities as well as practices analytical skills in the long term [40, 87, 184]. Therefore, one can imagine exploring a mixed approach by combining bottom-up and top-down reading and sensemaking (i.e., almost all the systems introduced in this thesis), leveraging the advantages of both. Below, we sketch out a possible variation that gradually increases the *grounding features provided by the system as users progress through their sensemaking stages.*

From an information gain perspective, the first web page a user reads on a topic provides valuable new information. Users can digest this information on their own to form a personalized mental model. In this initial phase, the system can, for example, compute common criteria in the background *without* presenting them or providing in-situ paragraph-level annotations. Instead, the criteria will only be revealed based on users' behavioral signals; for example, only when people pause to read a paragraph for a while (like in Crystalline) or explicitly collect information (like in Unakite and Wigglite) would the system augment that specific paragraph with phrase-level annotations. This approach preserves users' opportunity for bottom-up initial sensemaking while using annotations as a *a quality assurance* to help users *verify their self-constructed mental model in real time.* Henceforth, subsequent readings practically serve as updates to an existing mental model, reducing the risk of the full-on overview and annotations biasing users. In the meantime, the system can then function as a *coverage checker*, guiding users to continue finding, reading, and collecting information about unseen criteria in a top-down manner.

8.3 Future Work

I strongly believe that there is still a great deal of uncharted territory in the realm of sensemaking support. Below, I will briefly explore several promising avenues for future research.

8.3.1 Flexible collection and organization tools

Building on the foundation laid out by my existing work, I further recognize that the dynamic and evolving nature of sensemaking — particularly in the early stages — means that users often would avoid committing to a particular structure or doing any type of structuring at all. Additionally, prior work that provides structures using tables [166, 231, 232] and decision trees [269] have proven too rigid and constraining for the iterative and incremental process that people naturally use for sensemaking when investigating what decision to make. Therefore, future work should develop more flexible alternatives. For example, a starting point would be ordered lists of important criteria, where each can be marked as to whether it is a *deal-breaker* (if it does not have the appropriate value, then the option does not need to be further considered), *trade-off* (values in this criteria trade-off with values in another, and the user must balance which is more important), or *equivalence class* (all the options are pretty much the same for this criteria, so it turns out not to be a useful discriminator). Then, a table might be automatically created just for the trade-offs. We propose integrating other views, such as a map view for spatial constraints, and providing flexible, interactive transitions among views.

As mentioned above, another key goal will be easily *discarding* or *archiving* of extracted information if it is decided to be irrelevant or not useful. These interactions can take advantage of lightweight signals similar to those discussed in chapters 4 and 5. Future work can also investigate making it easy for users to change their minds. For example, users may decide that an archived item is actually still useful later. Similarly, if a user decides that a criterion is a deal-breaker, but then all viable options are eliminated, that classification may need to be revised, which will potentially have ripple effects on many structures. Here, prior research on *selective* undo [187, 374] could be of help, allowing users to easily find and change their prior actions without necessarily needing to undo changes that they still want to keep.

8.3.2 Opportunities for Collaborative Sensemaking

Through the years, people who are involved in decision-making processes, such as study participants and industry connections, would often suggest that the platform described in this thesis has the potential to become a collaborative platform for people to *cooperate* and *build on each other's*

decision-making processes. This is in line with our vision to add support for both asynchronous and synchronous collaborations in the future. Presently, my systems (Unakite, Crystalline, Wigglite, etc.) can help a user keep track of a static snapshot of a user's learning, sensemaking, and decision-making trails that are read-only to other people (through Strata). In future iterations, one can imagine mechanisms that enable subsequent people to "own" or "contribute" to a table that a previous person made (effectively creating new versions of that table without overriding the original author's version) so that it stays relevant and informative throughout the course of a project. For example, inspired by Git and other local version control tools such as Variolite [186], we can explore the opportunity of introducing lightweight versioning into Unakite, Crystalline, or Wigglite, thereby realizing asynchronous collaborations. Suggested by collaborative systems like SearchTogether [262] and CoSense [283], additional "awareness" and "division of labor" features can be implemented to transform my systems into a synchronous collaboration platform. Furthermore, it would also be an interesting challenge to aggregate the changes in different knowledge consumers' versions and propagate them back to the original author as constructive feedback.

8.3.3 Long-term impact on learning and decision-making

Automation afforded by systems like Crystalline and lightweight interactions afforded by systems like Wigglite enable people to focus their attention on reading and comprehending the web pages rather than splitting attention with having to collect and organize the information at the same time. In addition, augmented reading tools like Selenite further alleviate the burden of reading raw material by providing users with a synthesized and readily-digestible overview. However, prior work in learning science, such as Bransford et al. [87], found that people who *personally* performed the actions of collecting, categorizing, and organizing information were more likely to be able to recall it correctly and in detail, and exhibited increased confidence in the final outcome. This raises an interesting tension and trade-off between full-on automation and direct manipulation — future research would be required to examine the long-term effect on people's learning outcomes as well as confidence in their decisions using systems like Selenite, Crystalline, and Wigglite, and determine the appropriate levels and circumstances when automatic information synthesis and bookkeeping should be applied.

8.3.4 Implications on the future of generative AI-based search

While positioned as a grounded reading tool, my last piece of work, Selenite, also presents intriguing implications for the future design of AI-based search engines. As we transition from retrieval-based search to generation-based search, overcoming the obstacle of *LLM hallucination* [175] becomes crucial. Currently, commercial services like Bing Chat utilize large models to retrieve pertinent passages and generate abstractive summaries based on those sources. However, recent research shows that up to 50% of the references that Bing Chat presented at the end failed to substantiate its generated answers, and the current inline citation cannot effectively support users in verifying the search results [237]. Expanding on the earlier discussion about using Selenite for inline verification, one can imagine alternative designs for presenting generative AI search results. For example, alongside the text generated by the LLM, these models can also incorporate relevant paragraphs or phrases taken from the source pages as additional supporting evidence, increasing the perceived validity. In addition, signals can be given and collected in a lightweight way similar to that described in chapter 5 to provide feedback to the generative AI search engine for fast and modular improvement [38].

Moreover, while the current chat interface of LLM has shown success in various use cases, it may also disrupt the natural Google search workflow that users are familiar with. Google search gives users multiple ways and UI affordances that aid them in obtaining, comprehending, and validating information. For instance, users can read the paragraph that potentially contains the answer to their queries or explore related questions that other users have asked. Future research should investigate ways to integrate these familiar UI affordances and interactions into LLM's chat interface to enhance its overall usability and user experience, making the transition between traditional search and generative AI-based search more seamless.

8.4 Concluding Remarks

Modern search engines are undoubtedly valuable for finding information on the web. However, assembling this information into a useful mental model for learning and decision-making requires people to iteratively gather and synthesize information about available options and criteria. Unfortunately, existing tools like browser tabs, spreadsheets, and external note-taking apps poorly support the constant shifts between reading, collecting, extracting, organizing, and reorganizing that are essential in these sensemaking scenarios. Worse yet, even if people make the effort to externalize and share a summary of their sensemaking outcome, it can still be difficult for subse-

quent users to evaluate whether they can or should trust and reuse that work so that they don't have to start from scratch.

In this dissertation, I investigated the core thesis of providing effective and efficient tool support during sensemaking and decision-making by designing and building five novel systems and interaction techniques. The key insight is that users can benefit significantly from sensemaking systems that minimize interaction costs while offering appropriate augmentation. Through extensive performance and user evaluations, I demonstrated such benefits to users over existing approaches. For example, in Chapter 3, I showcased how the collection and organization of information can be streamlined and made in situ, and context switches between various tools and apps can be reduced. In Chapter 4, I explored the feasibility to harvest implicit human behaviors that people naturally exhibit when searching and browsing that reflect their mental models. In Chapter 5, I introduced intuitive wiggle-based gestures that simplify information collection and triaging tasks previously requiring multiple mouse clicks or actions. Additionally, in Chapter 7, I discussed the augmentation of people's reading and comprehension process with global overviews and in-context annotations. Finally, Chapter 6 addressed the issue of evaluating already-summarized knowledge for reuse by augmenting it with crucial signals that offer insights into its context, trustworthiness, and thoroughness.

The series of work introduced in this thesis points to the importance of having tool support that helps users efficiently organize and manage information as they find it in a way that could also be beneficial to others, taking a first step towards bootstrapping the virtuous cycle of people being able to build on each other's sensemaking results, fostering effective collaboration and knowledge reuse.

Appendix A

GPT-4 Prompts used in Selenite

Here, we outline the techniques we employed to guide the GPT-4 model, developed by OpenAI [274], within the context of Selenite. If not explicitly stated, the temperature of the model is set to 0.3 for a balance between consistency and creativity.

If not explicitly stated, the initial [System Message][1] was set as the following:

```
[System Message]
You are a helpful assistant that performs content analysis according to user re-
quests. Follow the user's requirements carefully and to the letter.
```

A.1 Getting topic from a web page

The prompt that we used to obtain a concise topic given a web page (with [title] and [content of the first few paragraphs]) is a two-step prompt:

```
[User Message] Step 1
Given the following information of an article:
Title: [title]
First few paragraphs: [content of the first few paragraphs]
What is this article about?

[Assistant Message]
This article is about ...
```

[1] The system message helps set the behavior of the model response, i.e., the [Assistant Message]. However, as stated by OpenAI: "... note that the system message is optional and the model's behavior without a system message is likely to be similar to using a generic message such as 'You are a helpful assistant' ..." (https://platform.openai.com/docs/guides/gpt/chat-completions-api)

[User Message] Step 2
I want to find articles similar to this one in terms of the general topic. What should I search for? Output one search phrase (in double quotes).

Additionally, we set the n parameter to 10, thereby instructing GPT-4 to produce 10 simultaneous responses. Subsequently, we determined the most commonly occurring one among these 10 as the **topic** for the article. While it could be assumed that a higher value for n would result in a lengthier response time from the model, our observations indicate that such delay is practically insignificant.

A.2 Getting options from a web page

The prompt that we used to obtain extract options from a given web page (with [title] and [content of the web page]) is a two-step prompt:

[User Message] Step 1
Given the following information of an article:

Title: [title]

Is the article likely to be discussing one or more aspects of "one specific option" (e.g., a single javascript framework, for example, React, or a single baby stroller option, or a specific Airbnb listing) or "multiple options/topics"? Output in the following format:
Reasoning: your reasoning process.
Verdict: "one specific option / multiple options"

[Assistant Message]
Reasoning: model's reasoning process...
Verdict: "one specific option / multiple options"

[User Message] Step 2
Now, given the content of the article below, what is/are the options?
Content:
[content of the web page]

```
Output should be in the following format: ["option_1", "option_2", ...]
```

A.3 Getting commonly-considered criteria from a web page

The prompt that we used to obtain a set of commonly considered criteria resembles something like the following (given a `[topic]`):

```
[User Message] Step 1: Ask for an initial set of criteria
What are some common aspects, criteria, or dimensions that people consider on
the topic of [topic]? Note that the criteria should be **most relevant to the
topic**, **frequently considered**, and can **cover a broad range of perspec-
tives**. Output should be a single bulleted list in the format of:

- Criterion: short description.

Do not output anything else.

[Assistant Message]
- [Criterion 1]: [Short description]
- [Criterion 2]: [Short description]
- [Criterion 3]: [Short description]
...

[User Message] Step 2+: Ask for additional criteria until we get around 20.
Give me five more that are different from, more diverse than, and possibly as
important as the ones listed above. Output in the same format.
```

A.4 Getting detailed analysis of text content

The prompts that we used to obtain a detailed analysis of text content given the `[text content]`, the list of `NLI criteria`, and the list of `[options]` on the corresponding web page is two-fold:

First, we ask GPT-4 to extract phrases from the content that describes a given criterion as well as determine each extracted phrase's sentiment with respect to the criterion:

```
[User Message]
Given the following **content** and list of **criteria**:

**Content**:

[content]

**Criteria (with definitions)**:
- [NLI Criterion 1]: [description]
- [NLI Criterion 2]: [description]
...

For each criterion: 1) extract **every possible** utterance that **mentions**
or **explicitly describes** that criterion from the content 2) perform senti-
ment analysis to determine if the utterance is "positive", "neutral", or "neg-
ative" with respect to that criterion. Remember to use the **exact same words**
from the content. Do not paraphrase!

Output must follow the format below:

## criterion_1_name
- "extracted_sentence_or_phrase_1" -> positive,
- "extracted_sentence_or_phrase_2" -> neutral,
## criterion_2_name
NONE FOUND
## criterion_3_name
- "extracted_sentence_or_phrase_1" -> neutral,
- "extracted_sentence_or_phrase_2" -> negative,
- "extracted_sentence_or_phrase_3" -> positive,
```

Second, we ask GPT-4 to label each extracted phrase with a possible [option] on the web page (we framed options as "subjects" of a phrase to achieve a better empirical performance):

```
[User Message]
Given the following **content** and the **phrases** extracted from the content
below:

**Content**:

[content]

**Extracted phrases**:
- "extracted_phrase_1"
- "extracted_phrase_2"
- "extracted_phrase_3"
...

For each phrase, determine the **subject** of the phrase based on the **content**.
Possible subjects are: [option_1, option_2, option_3, ...] Say "N/A" if you can-
not determine the subject. Output should be in the following format:

"extracted phrase 1" -> "subject" or "N/A"
"extracted phrase 2" -> "subject" or "N/A"
...
```

Bibliography

[1] Block-level elements - HTML: HyperText Markup Language | MDN. (????). https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements

[2] Build software better, together - Github. (????). https://github.com

[3] Getting started with machine learning. (????). https://github.com/collections/machine-learning

[4] npm | build amazing things. (????). https://www.npmjs.com/ Library Catalog: www.npmjs.com.

[5] Programming languages: A list of programming languages that are actively developed on GitHub. (????). https://github.com/collections/programming-languages

[6] Stack Overflow - Where Developers Learn, Share, & Build Careers. (????). https://stackoverflow.com/

[7] Wikipedia. (????). https://www.wikipedia.org/

[8] 2009a. PUT vs. POST in REST. (2009). https://stackoverflow.com/a/32524385

[9] 2009b. Which equals operator (== vs ===) should be used in JavaScript comparisons? (2009). https://stackoverflow.com/a/26923895

[10] 2018. *NumPy —NumPy*. http://www.numpy.org/

[11] 2019a. ARCore - Google Developer | ARCore. (2019). https://developers.google.com/ar/

[12] 2019b. ARKit - Apple Developer. (2019). https://developer.apple.com/arkit/

[13] 2019c. Front-end JavaScript frameworks. (2019). https://github.com/collections/front-end-javascript-frameworks

[14] 2019d. PyTorch. (2019). https://www.pytorch.org

[15] 2019e. TensorFlow. (2019). https://www.tensorflow.org/

[16] 2019f. ViroReact. (2019). https://viromedia.com/viroreact

[17] 2020a. "exports" config · Issue #20 · then/is-promise. (2020). https://github.com/then/is-promise/issues/20 Library Catalog: github.com.

[18] 2020b. pip - The Python Package Installer —pip 20.1 documentation. (2020). https://pip.pypa.io/en/stable/

[19] 2020c. Python Lists VS Numpy Arrays. (Feb. 2020). https://www.geeksforgeeks.org/python-lists-vs-numpy-arrays/ Library Catalog: www.geeksforgeeks.org Section: Python.

[20] 2020d. Stack Overflow Developer Survey 2020. (2020). https://insights.stackoverflow.com/survey/2020/

[21] 2021a. slick - the last carousel you'll ever need. (2021). http://kenwheeler.github.io/slick/

[22] 2021b. Splide - The lightweight, flexible and accessible slider/carousel. (2021). https://splidejs.com/

[23] 2021c. Swiper - The Most Modern Mobile Touch Slider. (2021). https://swiperjs.com/

[24] 2022. Vue.js. (2022). https://vuejs.org/

[25] B. Thomas Adler and Luca de Alfaro. 2007. A content-driven reputation system for the wikipedia. In *Proceedings of the 16th international conference on World Wide Web (WWW '07)*. Association for Computing Machinery, Banff, Alberta, Canada, 261–270. DOI:http://dx.doi.org/10.1145/1242572.1242608

[26] Janet E. Alexander and Marsha A. Tate. 1999. *Web Wisdom; How to Evaluate and Create Information Quality on the Webb* (1st ed.). L. Erlbaum Associates Inc., USA.

[27] Rana Alkadhi, Teodora Lata, Emitza Guzmany, and Bernd Bruegge. 2017. Rationale in Development Chat Messages: An Exploratory Study. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 436–446. DOI:http://dx.doi.org/10.1109/MSR.2017.43

[28] Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A Review on Language Models as Knowledge Bases. (April 2022). DOI:http://dx.doi.org/10.48550/arXiv.2204.06031 arXiv:2204.06031 [cs].

[29] Brian Amento, Loren Terveen, Will Hill, Deborah Hix, and Robert Schulman. 2003. Experiments in social data mining: The TopicShop system. *ACM Transactions on Computer-Human Interaction* 10, 1 (March 2003), 54–85. DOI:http://dx.doi.org/10.1145/606658.606661

[30] Saleema Amershi and Meredith Ringel Morris. 2008. CoSearch: A System for Co-located Collaborative Web Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1647–1656. DOI:http://dx.doi.org/10.1145/1357054.1357311

[31] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 3:1–3:13. DOI:http://dx.doi.org/10.1145/3290605.3300233 event-place: Glasgow, Scotland Uk.

[32] Jonathan Howard Amsbary and Larry Powell. 2003. Factors influencing evaluations of web site information. *Psychological Reports* 93, 1 (Aug. 2003), 191–198. DOI:http://dx.doi.org/10.2466/pr0.2003.93.1.191

[33] Jisun An, Daniele Quercia, and Jon Crowcroft. 2013. Why individuals seek diverse opinions (or why they don't). In *Proceedings of the 5th Annual ACM Web Science Conference (WebSci '13)*. Association for Computing Machinery, New York, NY, USA, 15–18. DOI:http://dx.doi.org/10.1145/2464464.2464493

[34] Jacqueline Anderson. 2009. Consumer Behavior Online: A 2009 Deep Dive. (2009). https://www.forrester.com/report/Consumer-Behavior-Online-A-2009-Deep-Dive/RES54327

[35] Apache. Apache Cordova. (????). https://cordova.apache.org/

[36] Kumaripaba Athukorala, Alan Medlar, Antti Oulasvirta, Giulio Jacucci, and Dorota Glowacka. 2016. Beyond Relevance: Adapting Exploration/Exploitation in Information Retrieval. In *Proceedings of the 21st International Conference on Intelligent User Interfaces (IUI '16)*. Association for Computing Machinery, New York, NY, USA, 359–369. DOI:http://dx.doi.org/10.1145/2856767.2856786

[37] Michelle Q. Wang Baldonado and Terry Winograd. 1997. SenseMaker: An Information-exploration Interface Supporting the Contextual Evolution of a User's Interests. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. ACM, New York, NY, USA, 11–18. DOI:http://dx.doi.org/10.1145/258549.258563

[38] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S. Weld, Walter S. Lasecki, and Eric Horvitz. 2019. Updates in Human-AI Teams: Understanding and Addressing the Performance/Compatibility Tradeoff. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (July 2019), 2429–2437. DOI:http://dx.doi.org/10.1609/aaai.v33i01.33012429 Number: 01.

[39] B.H. Barns and T.B. Bollinger. 1991. Making reuse cost-effective. *IEEE Software* 8, 1 (Jan. 1991), 13–24. DOI:http://dx.doi.org/10.1109/52.62928 Conference Name: IEEE Software.

[40] Marcia J Bates. 1989. The design of browsing and berrypicking techniques for the online search interface. *Online review* (1989). Publisher: MCB UP Ltd.

[41] David Bawden, Clive Holtham, and Nigel Courtney. 1999. Perspectives on information overload. *Aslib Proceedings* 51, 8 (Jan. 1999), 249–255. DOI:http://dx.doi.org/10.1108/EUM0000000006984 Publisher: MCB UP Ltd.

[42] David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, and Shilpa Dani. 2007. An engineering design knowledge reuse methodology using process modelling. *Research in Engineering Design* 18, 1 (May 2007), 37–48. DOI:http://dx.doi.org/10.1007/s00163-007-0028-8

[43] David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, and Shilpa Dani. 2008. A framework to integrate design knowledge reuse and requirements management in engineering design. *Robotics and Computer-Integrated Manufacturing* 24, 4 (Aug. 2008), 585–593. DOI:http://dx.doi.org/10.1016/j.rcim.2007.07.010

[44] Andrew Begel and Beth Simon. 2008. Novice software developers, all over again. In *Proceedings of the Fourth international Workshop on Computing Education Research (ICER '08)*. Association for Computing Machinery, Sydney, Australia, 3–14. DOI:http://dx.doi.org/10.1145/1404520.1404522

[45] Michael S. Bernstein, Jaime Teevan, Susan Dumais, Daniel Liebling, and Eric Horvitz. 2012. Direct Answers for Search Queries in the Long Tail. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 237–246. DOI:http://dx.doi.org/10.1145/2207676.2207710

[46] Krishna Bharat. 2000. SearchPad: explicit capture of search context to support Web search. *Computer Networks* 33, 1 (June 2000), 493–501. DOI:http://dx.doi.org/10.1016/S1389-1286(00)00047-5

[47] Eric A. Bier, Edward W. Ishak, and Ed Chi. 2006. Entity Workspace: An Evidence File That Aids Memory, Inference, and Reading. In *Intelligence and Security Informatics (Lecture Notes in Computer Science)*, Sharad Mehrotra, Daniel D. Zeng, Hsinchun Chen, Bhavani Thuraisingham, and Fei-Yue Wang (Eds.). Springer, Berlin, Heidelberg, 466–472. DOI:http://dx.doi.org/10.1007/11760146_42

[48] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. 1993. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93)*. Association for Computing Machinery, New York, NY, USA, 73–80. DOI:http://dx.doi.org/10.1145/166117.166126

[49] Jeffrey P. Bigham, Mingzhe Li, Samuel C. White, Xiaoyi Zhang, Qi Shan, and Carlos E. GUESTRIN. 2021. On-the-fly calibration for improved on-device eye tracking. (Aug. 2021). https://patents.google.com/patent/US11106280B1/en

[50] A.F. Blackwell. 2002. First Steps in Programming: A Rationale for Attention Investment Models. In *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments*. IEEE Comput. Soc, 2–10. DOI:http://dx.doi.org/10.1109/HCC.2002.1046334

[51] Pia Borlund. 2003. The concept of relevance in IR. *Journal of the American Society for Information Science and Technology* 54, 10 (2003), 913–925. DOI:http://dx.doi.org/10.1002/asi.10286 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.10286.

[52] Horatiu Bota, Paul N. Bennett, Ahmed Hassan Awadallah, and Susan T. Dumais. 2017. Self-Es: The Role of Emails-to-Self in Personal Information Management. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval (CHIIR '17)*. Association for Computing Machinery, New York, NY, USA, 205–214. DOI:http://dx.doi.org/10.1145/3020165.3020189

[53] D. Scott Brandt. 1996. Evaluating Information on the Internet. *Computers in Libraries* 16, 5 (1996), 44–46.

[54] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. 2010. Example-centric Programming: Integrating Web Search into the Development Environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 513–522. DOI:http://dx.doi.org/10.1145/1753326.1753402

[55] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1589–1598. DOI:http://dx.doi.org/10.1145/1518701.1518944 event-place: Boston, MA, USA.

[56] Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, Jennifer C Lai, and Robert L Mercer. 1992. An estimate of an upper bound for the entropy of English. *Computational Linguistics* 18, 1 (1992), 31–40.

[57] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. (July 2020). DOI:http://dx.doi.org/10.48550/arXiv.2005.14165 arXiv:2005.14165 [cs].

[58] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. 2019. Cross-Device Taxonomy: Survey, Opportunities and Challenges of Interactions Spanning Across Multiple Devices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–28. DOI:http://dx.doi.org/10.1145/3290605.3300792

[59] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. (April 2023). DOI:http://dx.doi.org/10.48550/arXiv.2303.12712 arXiv:2303.12712 [cs].

[60] Jürgen Buder, Christina Schwind, Anja Rudat, and Daniel Bodemer. 2015. Selective reading of large online forum discussions: The impact of rating visualizations on navigation and learning. *Computers in Human Behavior* 44 (March 2015), 191–201. DOI:http://dx.doi.org/10.1016/j.chb.2014.11.043

[61] Georg Buscher, Edward Cutrell, and Meredith Ringel Morris. 2009. What do you see when you're surfing? using eye tracking to predict salient regions of web pages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/1518701.1518705

[62] Georg Buscher, Andreas Dengel, and Ludger van Elst. 2008. Eye movements as implicit relevance feedback. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2991–2996. https://doi.org/10.1145/1358628.1358796

[63] Georg Buscher, Ludger van Elst, and Andreas Dengel. 2009. Segment-level display time as implicit feedback: a comparison to eye tracking. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 67–74. DOI:http://dx.doi.org/10.1145/1571941.1571955

[64] Vannevar Bush. 1945. As We May Think. (July 1945). http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/3881/ Section: Technology.

[65] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. 1988. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '88)*. Association for Computing Machinery, New York, NY, USA, 95–100. DOI:http://dx.doi.org/10.1145/57167.57182

[66] Robert Capra and Jaime Arguello. 2023. How does AI chat change search behaviors? (July 2023). DOI:http://dx.doi.org/10.48550/arXiv.2307.03826 arXiv:2307.03826 [cs].

[67] Rob Capra, Jaime Arguello, and Yinglong Zhang. 2017. The effects of search task determinability on search behavior. In *European Conference on Information Retrieval*. Springer, 108–121.

[68] Stuart K. Card, George G. Robertson, and William York. 1996. The WebBook and the Web Forager: Video Use Scenarios for a World-Wide Web Information Workspace. In *Conference Companion on Human Factors in Computing Systems (CHI '96)*. ACM, New York, NY, USA, 416–417. DOI:http://dx.doi.org/10.1145/257089.257407

[69] Nicholas J. Cepeda, Harold Pashler, Edward Vul, John T. Wixted, and Doug Rohrer. 2006. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin* 132, 3 (May 2006), 354–380. DOI:http://dx.doi.org/10.1037/0033-2909.132.3.354

[70] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Brussels, Belgium, 169–174. DOI:http://dx.doi.org/10.18653/v1/D18-2029

[71] Joseph Chee Chang, Nathan Hahn, Yongsung Kim, Julina Coupland, Bradley Breneisen, Hannah S Kim, John Hwong, and Aniket Kittur. 2021. When the Tab Comes Due:Challenges in the Cost Structure of Browser Tab Usage. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Number 148. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3411764.3445585

[72] Joseph Chee Chang, Nathan Hahn, and Aniket Kittur. 2016. Supporting Mobile Sensemaking Through Intentionally Uncertain Highlighting. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 61–68. DOI:http://dx.doi.org/10.1145/2984511.2984538

[73] Joseph Chee Chang, Nathan Hahn, and Aniket Kittur. 2020. Mesh: Scaffolding Comparison Tables for Online Decision Making. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 391–405. DOI:http://dx.doi.org/10.1145/3379337.3415865

[74] Joseph Chee Chang, Nathan Hahn, Adam Perer, and Aniket Kittur. 2019. SearchLens: composing and capturing complex user interests for exploratory search. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*. Association for Computing Machinery, Marina del Ray, California, 498–509. DOI:http://dx.doi.org/10.1145/3301275.3302321

[75] Joseph Chee Chang, Yongsung Kim, Victor Miller, Michael Xieyang Liu, Brad A Myers, and Aniket Kittur. 2021. Tabs.do: Task-Centric Browser Tab Management. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 663–676. https://doi.org/10.1145/3472749.3474777

[76] Yu-Wei Chao, Yunfan Liu, Xieyang Liu, Huayi Zeng, and Jia Deng. 2018. Learning to Detect Human-Object Interactions. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 381–389. DOI:http://dx.doi.org/10.1109/WACV.2018.00048

[77] Kathy Charmaz. 2006. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. SAGE. Google-Books-ID: 2ThdBAAAQBAJ.

[78] Chen Chen, Simon T. Perrault, Shengdong Zhao, and Wei Tsang Ooi. 2014. BezelCopy: an efficient cross-application copy-paste technique for touchscreen smartphones. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces (AVI '14)*. Association for Computing Machinery, New York, NY, USA, 185–192. DOI:http://dx.doi.org/10.1145/2598153.2598162

[79] Mon Chu Chen, John R. Anderson, and Myeong Ho Sohn. 2001. What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems (CHI EA '01)*. Association for Computing Machinery, New York, NY, USA, 281–282. DOI:http://dx.doi.org/10.1145/634067.634234

[80] Xiang'Anthony' Chen, Chien-Sheng Wu, Tong Niu, Wenhao Liu, and Caiming Xiong. 2022. Marvista: A Human-AI Collaborative Reading Tool. *arXiv preprint arXiv:2207.08401* (2022).

[81] Yan Chen, Jaylin Herskovitz, Walter S. Lasecki, and Steve Oney. 2020. Bashon: A Hybrid Crowd-Machine Workflow for Shell Command Synthesis. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–8. DOI:http://dx.doi.org/10.1109/VL/HCC50065.2020.9127248 ISSN: 1943-6106.

[82] Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S. Lasecki, and Steve Oney. 2017. Codeon: On-Demand Software Development Assistance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6220–6231. DOI:http://dx.doi.org/10.1145/3025453.3025972

[83] Yan Chen, Maulishree Pandey, Jean Y. Song, Walter S. Lasecki, and Steve Oney. 2020. Improving Crowd-Supported GUI Testing with Structural Guidance. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. DOI:http://dx.doi.org/10.1145/3313831.3376835

[84] Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In *Perspectives on socially shared cognition*. American Psychological Association, Washington, DC, US, 127–149. DOI:http://dx.doi.org/10.1037/10096-006

[85] Mark Claypool, Phong Le, Makoto Wased, and David Brown. 2001. Implicit interest indicators. In *Proceedings of the 6th international conference on Intelligent user interfaces (IUI '01)*. Association for Computing Machinery, New York, NY, USA, 33–40. DOI:http://dx.doi.org/10.1145/359784.359836

[86] A. Cockburn and J. Highsmith. 2001. Agile software development, the people factor. *Computer* 34, 11 (Nov. 2001), 131–133. DOI:http://dx.doi.org/10.1109/2.963450 Conference Name: Computer.

[87] National Research Council and others. 2000. *How people learn: Brain, mind, experience, and school: Expanded edition*. National Academies Press.

[88] Anita Crescenzi, Yuan Li, Yinglong Zhang, and Rob Capra. 2019. Towards Better Support for Exploratory Search through an Investigation of Notes-to-self and Notes-to-share. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 1093–1096. DOI:http://dx.doi.org/10.1145/3331184.3331309

[89] Dick Cunningham and Scott L. Shablak. 1975. Selective Reading Guide-O-Rama: The Content Teacher's Best Friend. *The Journal of Reading* (1975).

[90] Douglass R. Cutting, David R. Karger, and Jan O. Pedersen. 1993. Constant Interaction-time Scatter/Gather Browsing of Very Large Document Collections. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93)*. ACM, New York, NY, USA, 126–134. DOI:http://dx.doi.org/10.1145/160688.160706

[91] Richard L. Daft and Karl E. Weick. 1984. Toward a Model of Organizations as Interpretation Systems. *The Academy of Management Review* 9, 2 (April 1984), 284. DOI:http://dx.doi.org/10.2307/258441

[92] Fernando Das-Neves, Edward A. Fox, and Xiaoyan Yu. 2005. Connecting topics in document collections with stepping stones and pathways. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM '05)*. Association for Computing Machinery, New York, NY, USA, 91–98. DOI:http://dx.doi.org/10.1145/1099554.1099573

[93] Thomas H. Davenport, Sirkka L. Jarvenpaa, and Michael C. Beers. 1996. Improving Knowledge Work Processes. *Sloan management review* 37, 4 (1996), 53–65. https://dialnet.unirioja.es/servlet/articulo?codigo=2514140 Publisher: MIT press Section: Sloan management review.

[94] Sergio Cozzetti B. de Souza, Nicolas Anquetil, and Káthia M. de Oliveira. 2005. A study of the documentation essential to software maintenance. In *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information (SIGDOC '05)*. Association for Computing Machinery, New York, NY, USA, 68–75. DOI:http://dx.doi.org/10.1145/1085313.1085331

[95] David Dearman and Jeffery S. Pierce. 2008. It's on my other computer! computing with multiple devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. Association for Computing Machinery, New York, NY, USA, 767–776. DOI:http://dx.doi.org/10.1145/1357054.1357177

[96] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. DOI:http://dx.doi.org/10.1109/CVPR.2009.5206848 ISSN: 1063-6919.

[97] Peter Denning, Jim Horning, David Parnas, and Lauren Weinstein. 2005. Wikipedia risks. *Commun. ACM* 48, 12 (Dec. 2005), 152. DOI:http://dx.doi.org/10.1145/1101779.1101804

[98] Brenda Dervin. 1983. An overview of sense-making research concepts, methods, and results to date. (1983). http://www.worldcat.org/title/overview-of-sense-making-research-concepts-methods-and-results-to-date/oclc/733067203

[99] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]* (May 2019). http://arxiv.org/abs/1810.04805 arXiv: 1810.04805.

[100] Nancy M. Dixon. 2000. *Common Knowledge: How Companies Thrive by Sharing What They Know.* Harvard Business School Press, USA.

[101] Mira Dontcheva, Steven M. Drucker, Geraldine Wade, David Salesin, and Michael F. Cohen. 2006. Summarizing Personal Web Browsing Sessions. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*. ACM, New York, NY, USA, 115–124. DOI:http://dx.doi.org/10.1145/1166253.1166273

[102] Paul Dourish and Victoria Bellotti. 1992. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work (CSCW '92)*. Association for Computing Machinery, Toronto, Ontario, Canada, 107–114. DOI:http://dx.doi.org/10.1145/143457.143468

[103] Tore Dybå and Torgeir Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50, 9 (Aug. 2008), 833–859. DOI:http://dx.doi.org/10.1016/j.infsof.2008.01.006

[104] Dora Dzvonyar, Stephan Krusche, Rana Alkadhi, and Bernd Bruegge. 2016. Context-Aware User Feedback in Continuous Software Evolution. In *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*. 12–18. DOI:http://dx.doi.org/10.1109/CSED.2016.011

[105] Douglas C Engelbart. 1962. Augmenting human intellect: A conceptual framework. *Menlo Park, CA* (1962).

[106] Thomas Erickson and Wendy A. Kellogg. 2000. Social translucence: an approach to designing systems that support social processes. *ACM Transactions on Computer-Human Interaction* 7, 1 (March 2000), 59–83. DOI:http://dx.doi.org/10.1145/344949.345004

[107] Evernote. Best Note Taking App - Organize Your Notes with Evernote. (????). https://evernote.com

[108] Gunther Eysenbach and Christian Köhler. 2002. How do consumers search for and appraise health information on the world wide web? Qualitative study using focus groups, usability tests, and in-depth interviews. *BMJ (Clinical research ed.)* 324, 7337 (March 2002), 573–577. DOI:http://dx.doi.org/10.1136/bmj.324.7337.573

[109] Facebook. 2018. React - A JavaScript library for building user interfaces. (2018). https://reactjs.org/

[110] Mingming Fan, Zhen Li, and Franklin Mingzhe Li. 2020. Eyelid Gestures on Mobile Devices for People with Motor Impairments. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '20)*. Association for Computing Machinery, New York, NY, USA, 1–8. DOI:http://dx.doi.org/10.1145/3373625.3416987

[111] Mingming Fan, Zhen Li, and Franklin Mingzhe Li. 2021. Eyelid gestures for people with motor impairments. *Commun. ACM* 65, 1 (Dec. 2021), 108–115. DOI:http://dx.doi.org/10.1145/3498367

[112] David K. Farkas and Christopher Raleigh. 2013. Designing Documents for Selective Reading. *Information Design Journal* 20, 1 (Jan. 2013), 2–15. DOI:http://dx.doi.org/10.1075/idj.20.1.01far Publisher: John Benjamins.

[113] Guillaume Faure, Olivier Chapuis, and Nicolas Roussel. 2009. Power tools for copying and moving: useful stuff for your desktop. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1675–1678. DOI:http://dx.doi.org/10.1145/1518701.1518958

[114] Kristie Fisher, Scott Counts, and Aniket Kittur. 2012. Distributed Sensemaking: Improving Sensemaking by Leveraging the Efforts of Previous Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 247–256. DOI:http://dx.doi.org/10.1145/2207676.2207711

[115] Andrew J. Flanagin and Miriam J. Metzger. 2000. Perceptions of Internet Information Credibility. *Journalism & Mass Communication Quarterly* 77, 3 (Sept. 2000), 515–540. DOI:http://dx.doi.org/10.1177/107769900007700304 Publisher: SAGE Publications Inc.

[116] B. J. Fogg. 2002. Persuasive technology: using computers to change what we think and do. *Ubiquity* 2002, December (Dec. 2002), 5:2. DOI:http://dx.doi.org/10.1145/764008.763957

[117] B. J. Fogg and Hsiang Tseng. 1999. The elements of computer credibility. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '99)*. Association for Computing Machinery, Pittsburgh, Pennsylvania, USA, 80–87. DOI:http://dx.doi.org/10.1145/302979.303001

[118] Raymond Fok, Hita Kambhamettu, Luca Soldaini, Jonathan Bragg, Kyle Lo, Marti Hearst, Andrew Head, and Daniel S Weld. 2023. Scim: Intelligent Skimming Support for Scientific Papers. In *Proceedings of the 28th International Conference on Intelligent User Interfaces (IUI '23)*. Association for Computing Machinery, New York, NY, USA, 476–490. DOI:http://dx.doi.org/10.1145/3581641.3584034

[119] Andrew Forward and Timothy C. Lethbridge. 2002. The relevance of software documentation, tools and technologies: a survey. In *Proceedings of the 2002 ACM symposium on Document engineering (DocEng '02)*. Association for Computing Machinery, New York, NY, USA, 26–33. DOI:http://dx.doi.org/10.1145/585058.585065

[120] David Foster. 2020. The Google 'vs' Trick. (June 2020). https://medium.com/applied-data-science/the-google-vs-trick-618c8fd5359f

[121] Adam Fourney and Meredith Ringel Morris. 2013. Enhancing Technical Q&A Forums with CiteHistory. In *Seventh International AAAI Conference on Weblogs and Social Media*. https://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/6082

[122] William Frakes and Carol Terry. 1996. Software reuse: metrics and models. *Comput. Surveys* 28, 2 (June 1996), 415–435. DOI:http://dx.doi.org/10.1145/234528.234531

[123] W B Frakes and B A Nejmeh. 1986. Software reuse through information retrieval. *ACM SIGIR Forum* 21, 1-2 (Sept. 1986), 30–36. DOI:http://dx.doi.org/10.1145/24634.24636

[124] Shane Frederick, George Loewenstein, and Ted O'Donoghue. 2002. Time Discounting and Time Preference: A Critical Review. *Journal of Economic Literature* 40, 2 (June 2002), 351–401. DOI:http://dx.doi.org/10.1257/002205102320161311

[125] John W. Fritch and Robert L. Cromwell. 2001. Evaluating Internet resources: Identity, affiliation, and cognitive authority in a networked world. *Journal of the American Society for Information Science and Technology* 52, 6 (2001), 499–507. DOI:http://dx.doi.org/10.1002/asi.1081 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.1081.

[126] Dennis A. Gioia and Kumar Chittipeddi. 1991. Sensemaking and Sensegiving in Strategic Change Initiation. *Strategic Management Journal* 12, 6 (1991), 433–448. http://www.jstor.org/stable/2486479

[127] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. 2012. Comparative Evaluation of Javascript Frameworks. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12 Companion)*. ACM, New York, NY, USA, 513–514. DOI:http://dx.doi.org/10.1145/2187980.2188103

[128] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722 [cs, stat]* (Feb. 2014). http://arxiv.org/abs/1402.3722 arXiv: 1402.3722.

[129] Victor M. González, Gloria Mark, and Gloria Mark. 2004. "Constant, Constant, Multi-tasking Craziness": Managing Multiple Working Spheres. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 113–120. DOI:http://dx.doi.org/10.1145/985692.985707 event-place: Vienna, Austria.

[130] Google. Angular - Angular Routing. (????). https://angular.io/guide/routing-overview

[131] Google. Angular - Introduction to Angular animations. (????). https://angular.io/guide/animations

[132] Google. Angular - Validating form input. (????). https://angular.io/guide/form-validation

[133] Google. 2012. Google Notebook. (2012). https://www.google.com/googlenotebook/faq.html

[134] Google. 2019. Angular - One Framework. Mobile & Desktop. (2019). https://angular.io/

[135] Google. 2021. Cloud Natural Language. (2021). https://cloud.google.com/natural-language

[136] Nitesh Goyal and Susan R. Fussell. 2016. Effects of Sensemaking Translucence on Distributed Collaborative Analysis. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. Association for Computing Machinery, New York, NY, USA, 288–302. DOI: http://dx.doi.org/10.1145/2818048.2820071

[137] Nitesh Goyal, Gilly Leshed, and Susan R. Fussell. 2013. Leveraging partner's insights for distributed collaborative sensemaking. In *Proceedings of the 2013 conference on Computer supported cooperative work companion (CSCW '13)*. Association for Computing Machinery, New York, NY, USA, 15–18. DOI:http://dx.doi.org/10.1145/2441955.2441960

[138] Katja Grace, John Salvatier, Allan Dafoe, Baobao Zhang, and Owain Evans. 2018. Viewpoint: When Will AI Exceed Human Performance? Evidence from AI Experts. *Journal of Artificial Intelligence Research* 62 (July 2018), 729–754. DOI:http://dx.doi.org/10.1613/jair.1.11222

[139] Qi Guo and Eugene Agichtein. 2008. Exploring mouse movements for inferring query intent. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '08)*. Association for Computing Machinery, New York, NY, USA, 707–708. DOI:http://dx.doi.org/10.1145/1390334.1390462

[140] Qi Guo and Eugene Agichtein. 2010a. Ready to buy or just browsing? detecting web searcher goals from interaction data. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10)*. Association for Computing Machinery, New York, NY, USA, 130–137. DOI: http://dx.doi.org/10.1145/1835449.1835473

[141] Qi Guo and Eugene Agichtein. 2010b. Towards predicting web searcher gaze position from mouse movements. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 3601–3606. https://doi.org/10.1145/1753846.1754025

[142] Stefan Haefliger, Georg von Krogh, and Sebastian Spaeth. 2007. Code Reuse in Open Source Software. *Management Science* 54, 1 (Nov. 2007), 180–193. DOI:http://dx.doi.org/10.1287/mnsc.1070.0748 Publisher: INFORMS.

[143] Nathan Hahn. 2020. *In-Situ Sensemaking Support Systems*. Ph.D. Dissertation. Carnegie Mellon University. https://www.proquest.com/openview/02479b254f6324e4aa87b416645f7fc3/1?pq-origsite=gscholar&cbl=18750&diss=y

[144] Nathan Hahn, Joseph Chang, Ji Eun Kim, and Aniket Kittur. 2016. The Knowledge Accelerator: Big Picture Thinking in Small Pieces. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2258–2270. DOI:http://dx.doi.org/10.1145/2858036.2858364

[145] Nathan Hahn, Joseph Chee Chang, and Aniket Kittur. 2018. Bento Browser: Complex Mobile Search Without Tabs. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, Montreal QC, Canada, 251:1–251:12. DOI:http://dx.doi.org/10.1145/3173574.3173825

[146] Udo Hahn and Ulrich Reimer. 1999. Knowledge-based text summarization: Salience and generalization operators for knowledge base abstraction. *Advances in automatic text summarization* (1999), 215–232. Publisher: MIT Press, Cambridge, Mass.

[147] Dianne J. Hall and Robert A. Davis. 2007. Engaging multiple perspectives: A value-based decision-making model. *Decision Support Systems* 43, 4 (Aug. 2007), 1588–1604. DOI:http://dx.doi.org/10.1016/j.dss.2006.03.004

[148] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Advances in Psychology*, Peter A. Hancock and Najmedin Meshkati (Eds.). Human Mental Workload, Vol. 52. North-Holland, 139–183. DOI:http://dx.doi.org/10.1016/S0166-4115(08)62386-9

[149] Björn Hartmann, Mark Dhillon, and Matthew K. Chan. 2011. HyperSource: Bridging the Gap Between Source and Code-related Web Sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2207–2210. DOI:http://dx.doi.org/10.1145/1978942.1979263 event-place: Vancouver, BC, Canada.

[150] Andrew Head, Elena L. Glassman, Björn Hartmann, and Marti A. Hearst. 2018. Interactive Extraction of Examples from Existing Code. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173659

[151] Marti Hearst, Ame Elliott, Jennifer English, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. 2002. Finding the Flow in Web Site Search. *Commun. ACM* 45, 9 (Sept. 2002), 42–49. DOI:http://dx.doi.org/10.1145/567498.567525 Place: New York, NY, USA Publisher: Association for Computing Machinery.

[152] Marti A. Hearst. 2014. What's Missing from Collaborative Search? *Computer* 47, 3 (March 2014), 58–61. DOI:http://dx.doi.org/10.1109/MC.2014.77

[153] Marti A. Hearst and Chandu Karadi. 1997. Cat-a-Cone: an interactive interface for specifying searches and viewing retrieval results using a large category hierarchy. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '97)*. Association for Computing Machinery, New York, NY, USA, 246–255. DOI:http://dx.doi.org/10.1145/258525.258582

[154] Anneli Heimbürger, Paula Silvonen, and Caj Södergård. 2001. A framework for automatic combination of media contents by minimising information redundancy. Case: Integrated publishing in multimedia networks. In *International Conference on Electronic Publishing*.

[155] Lawrence J. Henschen and Julia C. Lee. 2009. Using Semantic-Level Tags in HTML/XML Documents. In *Universal Access in Human-Computer Interaction. Applications and Services (Lecture Notes in Computer Science)*, Constantine Stephanidis (Ed.). Springer, Berlin, Heidelberg, 683–692. DOI:http://dx.doi.org/10.1007/978-3-642-02713-0_72

[156] Tom-Michael Hesse, Veronika Lerche, Marcus Seiler, Konstantin Knoess, and Barbara Paech. 2016. Documented decision-making strategies and decision knowledge in open source projects: An empirical study on Firefox issue reports. *Information and Software Technology* 79 (Nov. 2016), 36–51. DOI:http://dx.doi.org/10.1016/j.infsof.2016.06.003

[157] Ron R. Hightower, Laura T. Ring, Jonathan I. Helfman, Benjamin B. Bederson, and James D. Hollan. 1998. Graphical Multiscale Web Histories: A Study of Padprints. In *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia : Links, Objects, Time and Space—structure in Hypermedia Systems: Links, Objects, Time and Space—structure in Hypermedia Systems (HYPERTEXT '98)*. ACM, New York, NY, USA, 58–65. DOI:http://dx.doi.org/10.1145/276627.276634

[158] Yoshinori Hijikata. 2004. Implicit user profiling for on demand relevance feedback. In *Proceedings of the 9th international conference on Intelligent user interfaces (IUI '04)*. Association for Computing Machinery, New York, NY, USA, 198–205. DOI:http://dx.doi.org/10.1145/964442.964480

[159] Ken Hinckley, Xiaojun Bi, Michel Pahud, and Bill Buxton. 2012. Informal Information Gathering Techniques for Active Reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1893–1896. DOI:http://dx.doi.org/10.1145/2207676.2208327 event-place: Austin, Texas, USA.

[160] Andrew Hogue and David Karger. 2005. Thresher: automating the unwrapping of semantic content from the World Wide Web. In *Proceedings of the 14th international conference on World Wide Web (WWW '05)*. Association for Computing Machinery, New York, NY, USA, 86–95. DOI:http://dx.doi.org/10.1145/1060745.1060762

[161] Lichan Hong, Ed H. Chi, Raluca Budiu, Peter Pirolli, and Les Nelson. 2008. SparTag.us: a low cost tagging system for foraging of web content. In *Proceedings of the working conference on Advanced visual interfaces (AVI '08)*. Association for Computing Machinery, New York, NY, USA, 65–72. DOI:http://dx.doi.org/10.1145/1385569.1385582

[162] Johan F. Hoorn and Teunis D. van Wijngaarden. 2010. Web Intelligence for the Assessment of Information Quality: Credibility, Correctness, and Readability. *Web Intelligence and Intelligent Agents* (March 2010). DOI: http://dx.doi.org/10.5772/8372 Publisher: IntechOpen.

[163] Amber Horvath, Michael Xieyang Liu, River Hendriksen, Connor Shannon, Emma Paterson, Kazi Jawad, Andrew Macvean, and Brad A Myers. 2022. Understanding How Programmers Can Use Annotations on Documentation. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–16. DOI:http://dx.doi.org/10.1145/3491102.3502095

[164] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '99)*. Association for Computing Machinery, New York, NY, USA, 159–166. DOI:http://dx.doi.org/10.1145/302979.303030

[165] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]* (April 2017). http://arxiv.org/abs/1704.04861 arXiv: 1704.04861.

[166] Jane Hsieh, Michael Xieyang Liu, Brad A. Myers, and Aniket Kittur. 2018. An Exploratory Study of Web Foraging to Understand and Support Programming Decisions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 305–306. DOI:http://dx.doi.org/10.1109/VLHCC.2018.8506517 ISSN: 1943-6092.

[167] Donghan Hu and Sang Won Lee. 2020. ScreenTrack: Using a Visual History of a Computer Screen to Retrieve Documents and Web Pages. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, Honolulu, HI, USA, 1–13. DOI:http://dx.doi.org/10.1145/3313831.3376753

[168] Jeff Huang, Thomas Lin, and Ryen W. White. 2012a. No search result left behind: branching behavior with browser tabs. In *Proceedings of the fifth ACM international conference on Web search and data mining - WSDM '12*. ACM Press, Seattle, Washington, USA, 203. DOI:http://dx.doi.org/10.1145/2124295.2124322

[169] Jeff Huang, Ryen W. White, Georg Buscher, and Kuansan Wang. 2012b. Improving searcher models using mouse cursor activity. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (SIGIR '12)*. Association for Computing Machinery, New York, NY, USA, 195–204. DOI: http://dx.doi.org/10.1145/2348283.2348313

[170] HuggingFace. 2023. Perplexity of fixed-length models. (2023). https://huggingface.co/docs/transformers/perplexity

[171] Robert F. Hurley and G. Tomas M. Hult. 1998. Innovation, Market Orientation, and Organizational Learning: An Integration and Empirical Examination. *Journal of Marketing* 62, 3 (July 1998), 42–54. DOI:http://dx.doi.org/10.1177/002224299806200303 Publisher: SAGE Publications Inc.

[172] Ionic. Cross-Platform Mobile App Development. (????). https://ionicframework.com/

[173] Shamsi T. Iqbal, Jaime Teevan, Dan Liebling, and Anne Loomis Thompson. 2018. Multitasking with Play Write, a Mobile Microproductivity Writing Tool. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. Association for Computing Machinery, New York, NY, USA, 411–422. DOI: http://dx.doi.org/10.1145/3242587.3242611

[174] Zachary Ives, Craig Knoblock, Steve Minton, Marie Jacob, Partha Talukdar, Rattapoom Tuchinda, Jose Luis Ambite, Maria Muslea, and Cenk Gazen. 2009. Interactive Data Integration through Smart Copy & Paste. *arXiv:0909.1769 [cs]* (Sept. 2009). http://arxiv.org/abs/0909.1769 arXiv: 0909.1769.

[175] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55, 12 (March 2023), 248:1–248:38. DOI:http://dx.doi.org/10.1145/3571730

[176] Haojian Jin, Swarun Kumar, and Jason Hong. 2020. Providing architectural support for building privacy-sensitive smart home applications. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers (UbiComp-ISWC '20)*. Association for Computing Machinery, New York, NY, USA, 212–217. DOI: http://dx.doi.org/10.1145/3410530.3414328

[177] Haojian Jin, Minyi Liu, Kevan Dodhia, Yuanchun Li, Gaurav Srivastava, Matthew Fredrikson, Yuvraj Agarwal, and Jason I. Hong. 2018. Why Are They Collecting My Data? Inferring the Purposes of Network Traffic in Mobile Apps. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (Dec. 2018), 173:1–173:27. DOI:http://dx.doi.org/10.1145/3287051

[178] Haojian Jin, Tetsuya Sakai, and Koji Yatani. 2014. ReviewCollage: a mobile interface for direct comparison using online reviews. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services (MobileHCI '14)*. Association for Computing Machinery, New York, NY, USA, 349–358. DOI:http://dx.doi.org/10.1145/2628363.2628373

[179] Harish Kandala, B. K. Tripathy, and K. Manoj Kumar. 2018. A Framework to Collect and Visualize User' s Browser History for Better User Experience and Personalized Recommendations. In *Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 1 (Smart Innovation, Systems and Technologies)*, Suresh Chandra Satapathy and Amit Joshi (Eds.). Springer International Publishing, Cham, 218–224. DOI: http://dx.doi.org/10.1007/978-3-319-63673-3_26

[180] Eser Kandogan and Ben Shneiderman. 1997. Elastic Windows: Evaluation of Multi-window Operations. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. ACM, New York, NY, USA, 250–257. DOI:http://dx.doi.org/10.1145/258549.258720

[181] Hyeonsu B. Kang, Joseph Chee Chang, Yongsung Kim, and Aniket Kittur. 2022. Threddy: An Interactive System for Personalized Thread-based Exploration and Organization of Scientific Literature. (Aug. 2022). DOI: http://dx.doi.org/10.1145/3526113.3545660 arXiv:2208.03455 [cs].

[182] David R. Karger and Dennis Quan. 2004. Haystack: a user interface for creating, browsing, and organizing arbitrary semistructured information. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*. Association for Computing Machinery, New York, NY, USA, 777–778. DOI:http://dx.doi.org/10.1145/985921.985931

[183] Melanie Kellar, Carolyn Watters, and Michael Shepherd. 2007. A field study characterizing Web-based information-seeking tasks. *Journal of the American Society for Information Science and Technology* 58, 7 (May 2007), 999–1018. DOI:http://dx.doi.org/10.1002/asi.20590

[184] Andruid Kerne, Andrew M Webb, Steven M Smith, Rhema Linder, Nic Lupfer, Yin Qu, Jon Moeller, and Sashikanth Damaraju. 2014. Using metrics of curation to evaluate information-based ideation. *ACM Transactions on Computer-Human Interaction (ToCHI)* 21, 3 (2014), 1–48. Publisher: ACM New York, NY, USA.

[185] Mik Kersten and Gail C. Murphy. 2006. Using Task Context to Improve Programmer Productivity. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT '06/FSE-14)*. ACM, New York, NY, USA, 1–11. DOI:http://dx.doi.org/10.1145/1181775.1181777 event-place: Portland, Oregon, USA.

[186] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1265–1276. DOI:http://dx.doi.org/10.1145/3025453.3025626

[187] Mary Beth Kery, Bonnie E. John, Patrick O'Flaherty, Amber Horvath, and Brad A. Myers. 2019. Towards Effective Foraging by Data Scientists to Find Past Analysis Choices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. DOI:http://dx.doi.org/10.1145/3290605.3300322

[188] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E. John, and Brad A. Myers. 2018. The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–11. DOI:http://dx.doi.org/10.1145/3173574.3173748

[189] Mohammad Kianpisheh, Franklin Mingzhe Li, and Khai N. Truong. 2019. Face Recognition Assistant for People with Visual Impairments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (Sept. 2019), 90:1–90:24. DOI:http://dx.doi.org/10.1145/3351248

[190] Jong Wook Kim, K. Selçuk Candan, and Junichi Tatemura. 2009. Efficient overlap and content reuse detection in blogs and online news articles. In *Proceedings of the 18th international conference on World wide web (WWW '09)*. Association for Computing Machinery, New York, NY, USA, 81–90. DOI:http://dx.doi.org/10.1145/1526709.1526721

[191] Paul A Kirschner, Simon J Buckingham-Shum, and Chad S Carr. 2012. *Visualizing argumentation: Software tools for collaborative and educational sense-making*. Springer Science & Business Media.

[192] Aniket Kittur, Andrew M. Peters, Abdigani Diriye, and Michael Bove. 2014. Standing on the Schemas of Giants: Socially Augmented Information Foraging. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 999–1010. DOI:http://dx.doi.org/10.1145/2531602.2531644

[193] Aniket Kittur, Andrew M. Peters, Abdigani Diriye, Trupti Telang, and Michael R. Bove. 2013. Costs and Benefits of Structured Information Foraging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2989–2998. DOI:http://dx.doi.org/10.1145/2470654.2481415

[194] Aniket Kittur, Bongwon Suh, and Ed H. Chi. 2008. Can you ever trust a wiki? impacting perceived trustworthiness in wikipedia. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work (CSCW '08)*. Association for Computing Machinery, San Diego, CA, USA, 477–480. DOI:http://dx.doi.org/10.1145/1460563.1460639

[195] G. Klein, B. Moon, and R. R. Hoffman. 2006. Making Sense of Sensemaking 1: Alternative Perspectives. *IEEE Intelligent Systems* 21, 4 (July 2006), 70–73. DOI:http://dx.doi.org/10.1109/MIS.2006.75

[196] Torkel Klingberg. 2009. *The Overflowing Brain: Information Overload and the Limits of Working Memory*. Oxford University Press, USA. Google-Books-ID: IxMSDAAAQBAJ.

[197] Amy J. Ko, Robert DeLine, and Gina Venolia. 2007. Information Needs in Collocated Software Development Teams. In *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 344–353.

[198] Amy J. Ko, Brad A. Myers, and Htet Htet Aung. 2004. Six Learning Barriers in End-User Programming Systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC '04)*. IEEE Computer Society, Washington, DC, USA, 199–206. DOI:http://dx.doi.org/10.1109/VLHCC.2004.47

[199] Professor of Management and Director at the Institute of Management Georg Von Krogh, Georg von Krogh, Associate Professor in the Faculty of Social Sciences and the Graduate School of International Corporate Strategy Kazuo Ichijo, Kazuo Ichijo, Ikujiro Nonaka, and Professor of Graduate School of International Corporate Strategy at Hitotsubashi University and the Xerox Distinguished Professor in Knowledge at Hass School of Business Ikujiro Nonaka. 2000. *Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation*. Oxford University Press, USA. Google-Books-ID: JVESDAAAQBAJ.

[200] Charles W. Krueger. 1992. Software reuse. *Comput. Surveys* 24, 2 (June 1992), 131–183. DOI:http://dx.doi.org/10.1145/130844.130856

[201] Sachin Kumar, Vidhisha Balachandran, Lucille Njoo, Antonios Anastasopoulos, and Yulia Tsvetkov. 2023. Language Generation Models Can Cause Harm: So What Can We Do About It? An Actionable Survey. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Dubrovnik, Croatia, 3299–3321. https://aclanthology.org/2023.eacl-main.241

[202] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems –A survey. *Knowledge-Based Systems* 123 (May 2017), 154–162. DOI:http://dx.doi.org/10.1016/j.knosys.2017.02.009

[203] Gordon Kurtenbach and William Buxton. 1993. The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. Association for Computing Machinery, New York, NY, USA, 482–487. DOI:http://dx.doi.org/10.1145/169059.169426

[204] Andrew Kuznetsov, Joseph Chee Chang, Nathan Hahn, Napol Rachatasumrit, Bradley Breneisen, Julina Coupland, and Aniket Kittur. 2022. Fuse: In-Situ Sensemaking Support in the Browser. (Aug. 2022). DOI: http://dx.doi.org/10.1145/3526113.3545693 arXiv:2208.14861 [cs].

[205] Edward Lank and Eric Saund. 2005. Sloppy selection: Providing an accurate interpretation of imprecise selection gestures. *Computers & Graphics* 29, 4 (Aug. 2005), 490–500. DOI:http://dx.doi.org/10.1016/j.cag.2005.05.003

[206] Thomas D LaToza, David Garlan, James D Herbsleb, and Brad A Myers. 2007. Program comprehension as fact finding. In *ESEC/FSE 2007: ACM SIGSOFT Symposium on the Foundations of Software Engineering*. 361–370.

[207] Thomas D. LaToza and Brad A. Myers. 2010. Hard-to-answer Questions About Code. In *Evaluation and Usability of Programming Languages and Tools (PLATEAU '10)*. ACM, New York, NY, USA, 8:1–8:6. DOI: http://dx.doi.org/10.1145/1937117.1937125

[208] Thomas D. LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining Mental Models: A Study of Developer Work Habits. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. ACM, New York, NY, USA, 492–501. DOI:http://dx.doi.org/10.1145/1134285.1134355

[209] John Lawrence, Jonas Malmsten, Andrey Rybka, Daniel Sabol, and Ken Triplin. 2017. Comparing TensorFlow Deep Learning Performance Using CPUs, GPUs, Local PCs and Cloud. *Publications and Research* (May 2017). https://academicworks.cuny.edu/bx_pubs/50

[210] Huy Viet Le, Sven Mayer, Maximilian Weiß, Jonas Vogelsang, Henrike Weingärtner, and Niels Henze. 2020. Shortcut Gestures for Mobile Text Editing on Fully Touch Sensitive Smartphones. *ACM Transactions on Computer-Human Interaction* 27, 5 (Aug. 2020), 33:1–33:38. DOI:http://dx.doi.org/10.1145/3396233

[211] K. Lei, Y. Ma, and Z. Tan. 2014. Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. In *2014 IEEE 17th International Conference on Computational Science and Engineering*. 661–668. DOI:http://dx.doi.org/10.1109/CSE.2014.142

[212] T.C. Lethbridge, J. Singer, and A. Forward. 2003. How software engineers use documentation: the state of the practice. *IEEE Software* 20, 6 (Nov. 2003), 35–39. DOI:http://dx.doi.org/10.1109/MS.2003.1241364 Conference Name: IEEE Software.

[213] James R. Lewis. 2018. The System Usability Scale: Past, Present, and Future. *International Journal of Human–Computer Interaction* 34, 7 (July 2018), 577–590. DOI:http://dx.doi.org/10.1080/10447318.2018.1455307 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10447318.2018.1455307.

[214] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *CoRR* abs/1910.13461 (2019). http://arxiv.org/abs/1910.13461 arXiv: 1910.13461.

[215] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[216] Franklin Mingzhe Li, Di Laura Chen, Mingming Fan, and Khai N. Truong. 2021. "I Choose Assistive Devices That Save My Face": A Study on Perceptions of Accessibility and Assistive Technology Use Conducted in China. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–14. DOI:http://dx.doi.org/10.1145/3411764.3445321

[217] Franklin Mingzhe Li, Michael Xieyang Liu, Yang Zhang, and Patrick Carrington. 2022. Freedom to Choose: Understanding Input Modality Preferences of People with Upper-body Motor Impairments for Activities of Daily Living. (July 2022). DOI:http://dx.doi.org/10.1145/3517428.3544814 arXiv:2207.04344 [cs].

[218] Mingzhe Li, Mingming Fan, and Khai N. Truong. 2017. BrailleSketch: A Gesture-based Text Input Method for People with Visual Impairments. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '17)*. Association for Computing Machinery, New York, NY, USA, 12–21. DOI:http://dx.doi.org/10.1145/3132525.3132528

[219] Tianyi Li, Yasmine Belghith, Chris North, and Kurt Luther. 2020a. CrowdTrace: Visualizing Provenance in Distributed Sensemaking. In *2020 IEEE Visualization Conference (VIS)*. 191–195. DOI:http://dx.doi.org/10.1109/VIS47514.2020.00045

[220] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, Denver, Colorado, USA, 6038–6049. DOI:http://dx.doi.org/10.1145/3025453.3025483

[221] Toby Jia-Jun Li, Jingya Chen, Brandon Canfield, and Brad A. Myers. 2020b. Privacy-Preserving Script Sharing in GUI-based Programming-by-Demonstration Systems. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1 (May 2020), 060:1–060:23. DOI:http://dx.doi.org/10.1145/3392869

[222] Toby Jia-Jun Li, Jingya Chen, Haijun Xia, Tom M. Mitchell, and Brad A. Myers. 2020. Multi-Modal Repairs of Conversational Breakdowns in Task-Oriented Dialogs. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 1094–1107. DOI:http://dx.doi.org/10.1145/3379337.3415820

[223] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Wanling Ding, Tom M. Mitchell, and Brad A. Myers. 2018. APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Natural Language Instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 105–114. DOI:http://dx.doi.org/10.1109/VLHCC.2018.8506506 ISSN: 1943-6106.

[224] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent that Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New Orleans, LA, USA, 577–589. DOI:http://dx.doi.org/10.1145/3332165.3347899

[225] Toby Jia-Jun Li and Oriana Riva. 2018. Kite: Building Conversational Bots from Mobile Apps. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '18)*. Association for Computing Machinery, Munich, Germany, 96–109. DOI:http://dx.doi.org/10.1145/3210240.3210339

[226] Yang Li. 2010. Protractor: a fast and accurate gesture recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2169–2172. DOI:http://dx.doi.org/10.1145/1753326.1753654

[227] Joseph CR Licklider. 1960. Man-computer symbiosis. *IRE transactions on human factors in electronics* 1 (1960), 4–11. Publisher: IEEE.

[228] Brian Y. Lim and Anind K. Dey. 2009. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th international conference on Ubiquitous computing (UbiComp '09)*. Association for Computing Machinery, Orlando, Florida, USA, 195–204. DOI:http://dx.doi.org/10.1145/1620545.1620576

[229] Brian Y. Lim and Anind K. Dey. 2010. Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM international conference on Ubiquitous computing (UbiComp '10)*. Association for Computing Machinery, Copenhagen, Denmark, 13–22. DOI:http://dx.doi.org/10.1145/1864349.1864353

[230] Michael Xieyang Liu, Shaun Burley, Emily Deng, Angelina Zhou, Aniket Kittur, and Brad A. Myers. 2018a. Supporting Knowledge Acceleration for Programming from a Sensemaking Perspective. *Sensemaking Workshop at CHI Conference on Human Factors in Computing Systems* (April 2018). https://par.nsf.gov/biblio/10152063-supporting-knowledge-acceleration-programming-from-sensemaking-perspective

[231] Michael Xieyang Liu, Nathan Hahn, Angelina Zhou, Shaun Burley, Emily Deng, Aniket Kittur, and Brad A. Myers. 2018b. UNAKITE: Support Developers for Capturing and Persisting Design Rationales When Solving Problems Using Web Resources. *Workshop on Designing Technologies to Support Human Problem Solving at the IEEE Symposium on Visual Languages and Human-Centric Computing* (Oct. 2018). https://par.nsf.gov/biblio/10152060-unakite-support-developers-capturing-persisting-design-rationales-when-solving-problems-u

[232] Michael Xieyang Liu, Jane Hsieh, Nathan Hahn, Angelina Zhou, Emily Deng, Shaun Burley, Cynthia Taylor, Aniket Kittur, and Brad A. Myers. 2019. Unakite: Scaffolding Developers' Decision-Making Using the Web. In *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. ACM, New Orleans, LA, USA, 67–80. DOI:http://dx.doi.org/10.1145/3332165.3347908 event-place: New Orleans, LA, USA.

[233] Michael Xieyang Liu, Aniket Kittur, and Brad A. Myers. 2021. To Reuse or Not To Reuse? A Framework and System for Evaluating Summarized Knowledge. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (April 2021), 166:1–166:35. DOI:http://dx.doi.org/10.1145/3449240

[234] Michael Xieyang Liu, Aniket Kittur, and Brad A. Myers. 2022a. Crystalline: Lowering the Cost for Developers to Collect and Organize Information for Decision Making. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA. DOI:http://dx.doi.org/10.1145/3491102.3501968 event-place: New Orleans, LA, USA.

[235] Michael Xieyang Liu, Andrew Kuznetsov, Yongsung Kim, Joseph Chee Chang, Aniket Kittur, and Brad A. Myers. 2022b. Wigglite: Low-cost Information Collection and Triage. In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22)*. Association for Computing Machinery, New York, NY, USA. DOI:http://dx.doi.org/10.1145/3526113.3545661

[236] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D. Gordon. 2023a. "What It Wants Me To Say" : Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–31. DOI:http://dx.doi.org/10.1145/3544548.3580817

[237] Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023b. Evaluating Verifiability in Generative Search Engines. (April 2023). DOI:http://dx.doi.org/10.48550/arXiv.2304.09848 arXiv:2304.09848 [cs].

[238] Ziming Liu. 2005. Reading behavior in the digital environment: Changes in reading behavior over the past ten years. *Journal of Documentation* 61, 6 (Jan. 2005), 700–712. DOI:http://dx.doi.org/10.1108/00220410510632040 Publisher: Emerald Group Publishing Limited.

[239] Yoelle S. Maarek, Michal Jacovi, Menachem Shtalhaim, Sigalit Ur, Dror Zernik, Israel Z. Ben-Shaul, Yoelle S. Maarek, Michal Jacovi, Menachem Shtalhaim, Sigalit Ur, Dror Zernik, and Israel Z. Ben-Shaul. 1997. WebCutter: a system for dynamic and tailorable site mapping. *Computer Networks and ISDN Systems* 29, 8-13 (Sept. 1997), 1269–1279. DOI:http://dx.doi.org/10.1016/S0169-7552(97)00050-0

[240] Jock D Mackinlay, Ramana Rao, and Stuart K Card. 1995. An organic user interface for searching citation links. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 67–73.

[241] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. (May 2023). DOI:http://dx.doi.org/10.48550/arXiv.2303.17651 arXiv:2303.17651 [cs].

[242] Ann Majchrzak, Lynne P. Cooper, and Olivia E. Neece. 2004. Knowledge Reuse for Innovation. *Management Science* 50, 2 (Feb. 2004), 174–188. DOI:http://dx.doi.org/10.1287/mnsc.1030.0116 Publisher: INFORMS.

[243] Alireza Mansouri, Lilly Suriani Affendey, and Ali Mamat. 2008. Named entity recognition approaches. *International Journal of Computer Science and Network Security* 8, 2 (2008), 339–344. Publisher: Citeseer.

[244] Gary Marchionini. 1995. *Information Seeking in Electronic Environments*. Cambridge University Press, Cambridge. DOI:http://dx.doi.org/10.1017/CBO9780511626388

[245] Gary Marchionini. 2006. Exploratory Search: From Finding to Understanding. *Commun. ACM* 49, 4 (April 2006), 41–46. DOI:http://dx.doi.org/10.1145/1121949.1121979

[246] Lynne M. Markus. 2001. Toward a Theory of Knowledge Reuse: Types of Knowledge Reuse Situations and Factors in Reuse Success. *Journal of Management Information Systems* 18, 1 (May 2001), 57–93. DOI:http://dx.doi.org/10.1080/07421222.2001.11045671 Publisher: Routledge _eprint: https://doi.org/10.1080/07421222.2001.11045671.

[247] Catherine C. Marshall and Sara Bly. 2005. Saving and Using Encountered Information: Implications for Electronic Periodicals. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 111–120. DOI:http://dx.doi.org/10.1145/1054972.1054989 event-place: Portland, Oregon, USA.

[248] Catherine C. Marshall and Frank M. Shipman. 1995. Spatial hypertext: designing for change. *Commun. ACM* 38, 8 (Aug. 1995), 88–97. DOI:http://dx.doi.org/10.1145/208344.208350

[249] Michael K. Martin, Juergen Pfeffer, and Kathleen M. Carley. 2013. Network text analysis of conceptual overlap in interviews, newspaper articles and keywords. *Social Network Analysis and Mining* 3, 4 (Dec. 2013), 1165–1177. DOI:http://dx.doi.org/10.1007/s13278-013-0129-5

[250] Joaquim Mendes, Nuno Laranjeiro, and Marco Vieira. 2018. Toward characterizing HTML defects on the Web. *Software: Practice and Experience* 48, 3 (2018), 750–757. DOI:http://dx.doi.org/10.1002/spe.2545 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2545.

[251] Marc Meola. 2004. Chucking the Checklist: A Contextual Approach to Teaching Undergraduates Web-Site Evaluation. *portal: Libraries and the Academy* 4, 3 (July 2004), 331–344. DOI:http://dx.doi.org/10.1353/pla.2004.0055 Publisher: Johns Hopkins University Press.

[252] Danaë Metaxa, Joon Sung Park, Ronald E. Robertson, Karrie Karahalios, Christo Wilson, Jeff Hancock, and Christian Sandvig. 2021. Auditing Algorithms: Understanding Algorithmic Systems from the Outside In. *Foundations and Trends® in Human–Computer Interaction* 14, 4 (Nov. 2021), 272–344. DOI:http://dx.doi.org/10.1561/1100000083 Publisher: Now Publishers, Inc.

[253] Miriam J. Metzger. 2007. Making sense of credibility on the Web: Models for evaluating online information and recommendations for future research. *Journal of the American Society for Information Science and Technology* 58, 13 (2007), 2078–2091. DOI:http://dx.doi.org/10.1002/asi.20672

[254] Miriam J. Metzger, Andrew J. Flanagin, Keren Eyal, Daisy R. Lemus, and Robert M. Mccann. 2003. Credibility for the 21st Century: Integrating Perspectives on Source, Message, and Media Credibility in the Contemporary Media Environment. *Annals of the International Communication Association* 27, 1 (Jan. 2003), 293–335. DOI:http://dx.doi.org/10.1080/23808985.2003.11679029 Publisher: Routledge _eprint: https://doi.org/10.1080/23808985.2003.11679029.

[255] Miriam J. Metzger, Andrew J. Flanagin, and Ryan B. Medders. 2010. Social and Heuristic Approaches to Credibility Evaluation Online. *Journal of Communication* 60, 3 (2010), 413–439. DOI:http://dx.doi.org/10.1111/j.1460-2466.2010.01488.x

[256] André N. Meyer, Thomas Fritz, Gail C. Murphy, and Thomas Zimmermann. 2014. Software Developers' Perceptions of Productivity. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*. ACM, New York, NY, USA, 19–29. DOI:http://dx.doi.org/10.1145/2635868.2635892 event-place: Hong Kong, China.

[257] Microsoft. 2023. Your AI-Powered Copilot for the Web. (2023). https://www.microsoft.com/en-us/bing

[258] Frances J. Milliken. 1990. Perceiving and Interpreting Environmental Change: An Examination of College Administrators' Interpretation of Changing Demographics. *Academy of Management* 33, 1 (March 1990), 42–63. DOI:http://dx.doi.org/10.2307/256351

[259] Audris Mockus. 2007. Large-Scale Code Reuse in Open Source Software. In *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*. 7–7. DOI:http://dx.doi.org/10.1109/FLOSS.2007.10

[260] Dan Morris, Meredith Ringel Morris, and Gina Venolia. 2008. SearchBar: A Search-centric Web History for Task Resumption and Information Re-finding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1207–1216. DOI:http://dx.doi.org/10.1145/1357054.1357242

[261] M. R. Morris, A. J. B. Brush, and B. R. Meyers. 2007. Reading Revisited: Evaluating the Usability of Digital Display Surfaces for Active Reading Tasks. In *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07)*. 79–86. DOI:http://dx.doi.org/10.1109/TABLETOP.2007.12

[262] Meredith Ringel Morris and Eric Horvitz. 2007. SearchTogether: An Interface for Collaborative Web Search. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 3–12. DOI:http://dx.doi.org/10.1145/1294211.1294215

[263] Mozilla. 2022. Browser Extensions | MDN. (2022). https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions

[264] Sougata Mukherjea and James D. Foley. 1995. Visualizing the World-Wide Web with the Navigational View Builder. *Computer Networks and ISDN Systems* 27, 6 (April 1995), 1075–1087. DOI:http://dx.doi.org/10.1016/0169-7552(95)00023-Z

[265] B. Myers, R. Malkin, M. Bett, A. Waibel, B. Bostwick, R.C. Miller, Jie Yang, M. Denecke, E. Seemann, Jie Zhu, Choon Hong Peck, D. Kong, J. Nichols, and B. Scherlis. 2002. Flexi-modal and multi-machine user interfaces. In *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*. 343–348. DOI:http://dx.doi.org/10.1109/ICMI.2002.1167019

[266] Brad A. Myers, Amy J. Ko, Chris Scaffidi, Stephen Oney, YoungSeok Yoon, Kerry Chang, Mary Beth Kery, and Toby Jia-Jun Li. 2017. Making End User Development More Natural. In *New Perspectives in End-User Development*, Fabio Paternò and Volker Wulf (Eds.). Springer International Publishing, Cham, 1–22. DOI:http://dx.doi.org/10.1007/978-3-319-60291-2_1

[267] Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 5356–5371. DOI:http://dx.doi.org/10.18653/v1/2021.acl-long.416

[268] Theodor H Nelson. 1965. Complex information processing: a file structure for the complex, the changing and the indeterminate. In *Proceedings of the 1965 20th national conference*. 84–100.

[269] Phong H. Nguyen, Kai Xu, Andy Bardill, Betul Salman, Kate Herd, and B.L. William Wong. 2016. SenseMap: Supporting browser-based online sensemaking through analytic provenance. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 91–100. DOI:http://dx.doi.org/10.1109/VAST.2016.7883515

[270] Ikujiro Nonaka, Hirotaka Takeuchi, and Katsuhiro Umemoto. 1996. A theory of organizational knowledge creation. *International Journal of Technology Management* 11, 7-8 (Jan. 1996), 833–845. DOI:http://dx.doi.org/10.1504/IJTM.1996.025472 Publisher: Inderscience Publishers.

[271] Obsidian. 2022. Obsidian. (2022). https://obsidian.md/

[272] Carla O'Dell and C. Jackson Grayson. 1998. If Only We Knew What We Know: Identification and Transfer of Internal Best Practices:. *California Management Review* (April 1998). DOI:http://dx.doi.org/10.2307/41165948 Publisher: SAGE PublicationsSage CA: Los Angeles, CA.

[273] Stephen Oney and Joel Brandt. 2012. Codelets: Linking Interactive Documentation and Example Code in the Editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2697–2706. DOI:http://dx.doi.org/10.1145/2207676.2208664

[274] OpenAI. 2023. GPT-4 Technical Report. (March 2023). DOI:http://dx.doi.org/10.48550/arXiv.2303.08774 arXiv:2303.08774 [cs].

[275] Margit Osterloh and Bruno S. Frey. 2000. Motivation, Knowledge Transfer, and Organizational Forms. *Organization Science* 11, 5 (Oct. 2000), 538–550. DOI:http://dx.doi.org/10.1287/orsc.11.5.538.15204 Publisher: INFORMS.

[276] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. (March 2022). DOI:http://dx.doi.org/10.48550/arXiv.2203.02155 arXiv:2203.02155 [cs].

[277] Srishti Palani, Zijian Ding, Austin Nguyen, Andrew Chuang, Stephen MacNeil, and Steven P. Dow. 2021. CoNotate: Suggesting Queries Based on Notes Promotes Knowledge Discovery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–14. DOI:http://dx.doi.org/10.1145/3411764.3445618

[278] Alexandra Papoutsaki, James Laskey, and Jeff Huang. 2017. SearchGazer: Webcam Eye Tracking for Remote Studies of Web Search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval (CHIIR '17)*. Association for Computing Machinery, New York, NY, USA, 17–26. DOI:http://dx.doi.org/10.1145/3020165.3020170

[279] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, and James Hays. 2016. WebGazer: Scalable Webcam Eye Tracking Using User Interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI, 3839–3845.

[280] Soya Park, Amy X. Zhang, and David R. Karger. 2018. Post-literate Programming: Linking Discussion and Code in Software Development Teams. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings (UIST '18 Adjunct)*. ACM, New York, NY, USA, 51–53. DOI:http://dx.doi.org/10.1145/3266037.3266098 event-place: Berlin, Germany.

[281] Priyadarshini Patil, Prashant Narayankar, Narayan D.G., and Meena S.M. 2016. A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish. *Procedia Computer Science* 78 (Jan. 2016), 617–624. DOI:http://dx.doi.org/10.1016/j.procs.2016.02.108

[282] Emily S. Patterson and David D. Woods. 2001. Shift Changes, Updates, and the On-Call Architecture in Space Shuttle Mission Control. *Computer Supported Cooperative Work* 10, 3-4 (Dec. 2001), 317–346. DOI:http://dx.doi.org/10.1023/A:1012705926828

[283] Sharoda A. Paul and Meredith Ringel Morris. 2009. CoSense: Enhancing Sensemaking for Collaborative Web Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1771–1780. DOI:http://dx.doi.org/10.1145/1518701.1518974

[284] Sharoda A. Paul and Meredith Ringel Morris. 2011. Sensemaking in Collaborative Web Search. *Human–Computer Interaction* 26, 1-2 (March 2011), 72–122. DOI:http://dx.doi.org/10.1080/07370024.2011.559410

[285] Ksenia Peguero, Nan Zhang, and Xiuzhen Cheng. 2018. An Empirical Study of the Framework Impact on the Security of JavaScript Web Applications. In *Companion Proceedings of the The Web Conference 2018 (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 753–758. DOI:http://dx.doi.org/10.1145/3184558.3188736 event-place: Lyon, France.

[286] Pinterest. Pinterest. (????). https://www.pinterest.com/

[287] Peter Pirolli and Stuart Card. 1999. Information foraging. *Psychological Review* 106, 4 (1999), 643–675. DOI:http://dx.doi.org/10.1037/0033-295X.106.4.643 Place: US Publisher: American Psychological Association.

[288] Peter Pirolli and Stuart Card. 2005. The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. In *Proceedings of International Conference on Intelligence Analysis*. http://www.phibetaiota.net/wp-content/uploads/2014/12/Sensemaking-Process-Pirolli-and-Card.pdf

[289] Marlene A. Plumlee. 2003. The Effect of Information Complexity on Analysts' Use of That Information. *The Accounting Review* 78, 1 (Jan. 2003), 275–296. DOI:http://dx.doi.org/10.2308/accr.2003.78.1.275

[290] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. 2013. Seahawk: Stack Overflow in the IDE. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, San Francisco, CA, USA, 1295–1298. DOI:http://dx.doi.org/10.1109/ICSE.2013.6606701

[291] Luca Ponzanelli, Simone Scalabrino, Gabriele Bavota, Andrea Mocci, Rocco Oliveto, Massimiliano Di Penta, and Michele Lanza. 2017. Supporting Software Developers with a Holistic Recommender System. In *Proceedings of the 39th International Conference on Software Engineering (ICSE '17)*. IEEE Press, Piscataway, NJ, USA, 94–105. DOI:http://dx.doi.org/10.1109/ICSE.2017.17

[292] Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. 2014. A rule-based approach to aspect extraction from product reviews. In *Proceedings of the second workshop on natural language processing for social media (SocialNLP)*. 28–37.

[293] Napol Rachatasumrit, Gonzalo Ramos, Jina Suh, Rachel Ng, and Christopher Meek. 2021. ForSense: Accelerating Online Research Through Sensemaking Integration and Machine Research Support. In *26th International Conference on Intelligent User Interfaces (IUI '21)*. Association for Computing Machinery, New York, NY, USA, 608–618. DOI:http://dx.doi.org/10.1145/3397481.3450649

[294] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and others. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[295] Paruj Ratanaworabhan, Benjamin Livshits, and Benjamin G. Zorn. 2010. JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications. In *Proceedings of the 2010 USENIX Conference on Web Application Development (WebApps'10)*. USENIX Association, Berkeley, CA, USA, 3–3. http://dl.acm.org/citation.cfm?id=1863166.1863169 event-place: Boston, MA.

[296] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. (Aug. 2019). DOI:http://dx.doi.org/10.48550/arXiv.1908.10084 arXiv:1908.10084 [cs].

[297] John Rieman. 1993. The diary study: a workplace-oriented research tool to guide laboratory efforts. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. Association for Computing Machinery, New York, NY, USA, 321–326. DOI:http://dx.doi.org/10.1145/169059.169255

[298] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. 1998. Data mountain: using spatial memory for document management. In *Proceedings of the 11th annual ACM symposium on User interface software and technology (UIST '98)*. Association for Computing Machinery, New York, NY, USA, 153–162. DOI:http://dx.doi.org/10.1145/288392.288596

[299] Kerry Rodden, Xin Fu, Anne Aula, and Ian Spiro. 2008. Eye-mouse coordination patterns on web search results pages. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2997–3002. https://doi.org/10.1145/1358628.1358797

[300] Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. 2012. How Do Professional Developers Comprehend Software?. In *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*. IEEE Press, Piscataway, NJ, USA, 255–265. http://dl.acm.org/citation.cfm?id=2337223.2337254

[301] Daniel E. Rose and Danny Levinson. 2004. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web (WWW '04)*. Association for Computing Machinery, New York, NY, USA, 13–19. DOI:http://dx.doi.org/10.1145/988672.988675

[302] Mary Beth Rosson and John M. Carroll. 1996. The Reuse of Uses in Smalltalk Programming. *ACM Trans. Comput.-Hum. Interact.* 3, 3 (Sept. 1996), 219–253. DOI:http://dx.doi.org/10.1145/234526.234530

[303] Volker Roth and Thea Turner. 2009. Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1523–1526. DOI:http://dx.doi.org/10.1145/1518701.1518933

[304] Nirmal Roy, Manuel Valle Torre, Ujwal Gadiraju, David Maxwell, and Claudia Hauff. 2021. Note the Highlight: Incorporating Active Reading Tools in a Search as Learning Environment. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval (CHIIR '21)*. Association for Computing Machinery, New York, NY, USA, 229–238. DOI:http://dx.doi.org/10.1145/3406522.3446025

[305] Dean Rubine. 1991. Specifying gestures by example. *ACM SIGGRAPH Computer Graphics* 25, 4 (July 1991), 329–337. DOI:http://dx.doi.org/10.1145/127719.122753

[306] Dean Rubine. 1992. Combining gestures and direct manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. Association for Computing Machinery, New York, NY, USA, 659–660. DOI:http://dx.doi.org/10.1145/142750.143072

[307] Daniel M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. 1993. The Cost Structure of Sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. ACM, New York, NY, USA, 269–276. DOI:http://dx.doi.org/10.1145/169059.169209

[308] N. Rutar, C. B. Almazan, and J. S. Foster. 2004. A comparison of bug finding tools for Java. In *15th International Symposium on Software Reliability Engineering*. 245–256. DOI:http://dx.doi.org/10.1109/ISSRE.2004.1

[309] Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *Proceedings of the 19th international conference on World wide web (WWW '10)*. Association for Computing Machinery, Raleigh, North Carolina, USA, 841–850. DOI:http://dx.doi.org/10.1145/1772690.1772776

[310] Caitlin Sadowski, Kathryn T. Stolee, and Sebastian Elbaum. 2015. How Developers Search for Code: A Case Study. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*. ACM, New York, NY, USA, 191–201. DOI:http://dx.doi.org/10.1145/2786805.2786855

[311] Tefko Saracevic. Relevance reconsidered.

[312] Bill N. Schilit, Gene Golovchinsky, and Morgan N. Price. 1998. Beyond paper: supporting active reading with free form digital ink annotations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '98)*. ACM Press/Addison-Wesley Publishing Co., USA, 249–256. DOI:http://dx.doi.org/10.1145/274644.274680

[313] Ann Scholz-Crane. 1998. Evaluating the Future: A Preliminary Study of the Process of How Undergraduate Students Evaluate Web Sources. *RSR: Reference Services Review* 26 (1998), 53–60.

[314] M. C. Schraefel and Yuxiang Zhu. 2001. Interaction design for Web-based, within-page collection making and management. In *Proceedings of the 12th ACM conference on Hypertext and Hypermedia (HYPERTEXT '01)*. Association for Computing Machinery, New York, NY, USA, 125. DOI:http://dx.doi.org/10.1145/504216.504247

[315] M. C. schraefel, Yuxiang Zhu, David Modjeska, Daniel Wigdor, and Shengdong Zhao. 2002. Hunter Gatherer: Interaction Support for the Creation and Management of Within-web-page Collections. In *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*. ACM, New York, NY, USA, 172–181. DOI:http://dx.doi.org/10.1145/511446.511469

[316] Rever Score. 2017. Why we moved from Angular 2 to Vue.js (and why we didn' t choose React). (Sept. 2017). https://medium.com/reverdev/why-we-moved-from-angular-2-to-vue-js-and-why-we-didnt-choose-react-ef807d9f4163 Library Catalog: medium.com.

[317] Mirjam Seckler, Silvia Heinz, Seamus Forde, Alexandre N. Tuch, and Klaus Opwis. 2015. Trust and distrust on the web: User experiences and website characteristics. *Computers in Human Behavior* 45 (April 2015), 39–50. DOI:http://dx.doi.org/10.1016/j.chb.2014.11.064

[318] Nikhil Sharma. 2008. Sensemaking handoff: When and how? *Proceedings of the American Society for Information Science and Technology* 45, 1 (Jan. 2008), 1–12. DOI:http://dx.doi.org/10.1002/meet.2008.1450450234

[319] Nikhil Sharma. 2011. Role of available and provided resources in sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, Vancouver, BC, Canada, 1807–1816. DOI:http://dx.doi.org/10.1145/1978942.1979204

[320] Nikhil Sharma and George Furnas. 2009. Artifact usefulness and usage in sensemaking handoffs. *Proceedings of the American Society for Information Science and Technology* 46 (2009). DOI:http://dx.doi.org/10.1002/meet.2009.1450460219

[321] Johanna Shelby and Robert Capra. 2011. Sensemaking in collaborative exploratory search. *Proceedings of the American Society for Information Science and Technology* 48, 1 (2011), 1–3. DOI:http://dx.doi.org/10.1002/meet.2011.14504801318

[322] B. Shneiderman. 1996. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*. 336–343. DOI:http://dx.doi.org/10.1109/VL.1996.545307 ISSN: 1049-2615.

[323] Ben Shneiderman. 2000. Designing trust into online experiences. *Commun. ACM* 43, 12 (Dec. 2000), 57–59. DOI:http://dx.doi.org/10.1145/355112.355124

[324] Ben Shneiderman. 2020. Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy. *International Journal of Human–Computer Interaction* 36, 6 (April 2020), 495–504. DOI: http://dx.doi.org/10.1080/10447318.2020.1741118 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10447318.2020.1741118.

[325] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. 2006. Questions Programmers Ask During Software Evolution Tasks. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT '06/FSE-14)*. ACM, New York, NY, USA, 23–34. DOI:http://dx.doi.org/10.1145/1181775.1181779

[326] Amit Singhal and others. 2001. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.* 24, 4 (2001), 35–43.

[327] Manuel Sojer and Joachim Henkel. 2010. *Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments*. SSRN Scholarly Paper ID 1489789. Social Science Research Network, Rochester, NY. https://papers.ssrn.com/abstract=1489789

[328] Jean Y. Song, Stephan J. Lemmer, Michael Xieyang Liu, Shiyan Yan, Juho Kim, Jason J. Corso, and Walter S. Lasecki. 2019. Popup: reconstructing 3D video using particle filtering to aggregate crowd responses. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*. Association for Computing Machinery, Marina del Ray, California, 558–569. DOI:http://dx.doi.org/10.1145/3301275.3302305

[329] J Stylos and Brad A. Myers. 2006. Mica: A Web-Search Tool for Finding API Components and Examples. In *Visual Languages and Human-Centric Computing (VL/HCC'06)*. 195–202.

[330] Jeffrey Stylos, Brad A. Myers, and Andrew Faulring. 2004. Citrine: providing intelligent copy-and-paste. In *Proceedings of the 17th annual ACM symposium on User interface software and technology (UIST '04)*. Association for Computing Machinery, New York, NY, USA, 185–188. DOI:http://dx.doi.org/10.1145/1029632.1029665

[331] Atsushi Sugiura and Yoshiyuki Koseki. 1998. Internet scrapbook: automating Web browsing tasks by demonstration. In *Proceedings of the 11th annual ACM symposium on User interface software and technology (UIST '98)*. Association for Computing Machinery, New York, NY, USA, 9–18. DOI:http://dx.doi.org/10.1145/288392.288395

[332] Bongwon Suh, Ed H. Chi, Aniket Kittur, and Bryan A. Pendleton. 2008. Lifting the veil: improving accountability and social transparency in Wikipedia with wikidashboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. Association for Computing Machinery, Florence, Italy, 1037–1040. DOI:http://dx.doi.org/10.1145/1357054.1357214

[333] Wei Sun, Franklin Mingzhe Li, Benjamin Steeper, Songlin Xu, Feng Tian, and Cheng Zhang. 2021. TeethTap: Recognizing Discrete Teeth Gestures Using Motion and Acoustic Sensing on an Earpiece. In *26th International Conference on Intelligent User Interfaces (IUI '21)*. Association for Computing Machinery, New York, NY, USA, 161–169. DOI:http://dx.doi.org/10.1145/3397481.3450645

[334] Apple Support. 2022. Make the pointer easier to see on Mac. (2022). https://support.apple.com/guide/mac-help/make-the-pointer-easier-to-see-mchlp2920/12.0/mac/12.0

[335] Amanda Swearngin, Shamsi Iqbal, Victor Poznanski, Mark Encarnación, Paul N. Bennett, and Jaime Teevan. 2021. Scraps: Enabling Mobile Capture, Contextualization, and Use of Document Resources. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–14. DOI:http://dx.doi.org/10.1145/3411764.3445185

[336] João Sá, Vanessa Queiroz Marinho, Ana Rita Magalhães, Tiago Lacerda, and Diogo Goncalves. 2022. Diversity Vs Relevance: A Practical Multi-objective Study in Luxury Fashion Recommendations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2405–2409. DOI:http://dx.doi.org/10.1145/3477495.3531866

[337] Craig S. Tashman and W. Keith Edwards. 2011. Active reading and its discontents: the situations, problems and ideas of readers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2927–2936. DOI:http://dx.doi.org/10.1145/1978942.1979376

[338] Carol Tenopir, Donald W. King, Sheri Edwards, and Lei Wu. 2009. Electronic journals and changes in scholarly article seeking and reading patterns. *Aslib Proceedings* 61, 1 (Jan. 2009), 5–32. DOI:http://dx.doi.org/10.1108/00012530910932267 Publisher: Emerald Group Publishing Limited.

[339] Loren Terveen, Will Hill, and Brian Amento. 1999. Constructing, organizing, and visualizing collections of topically related Web resources. *ACM Transactions on Computer-Human Interaction* 6, 1 (March 1999), 67–94. DOI:http://dx.doi.org/10.1145/310641.310644

[340] Yi Yi Thaw, Ahmad Kamil Mahmood, and P. Dhanapal Durai Dominic. 2009. A Study on the Factors That Influence the Consumers Trust on Ecommerce Adoption. *arXiv:0909.1145 [cs]* (Sept. 2009). http://arxiv.org/abs/0909.1145 arXiv: 0909.1145.

[341] Meinald T. Thielsch and Gerrit Hirschfeld. 2019. Facets of Website Content. *Human–Computer Interaction* 34, 4 (July 2019), 279–327. DOI:http://dx.doi.org/10.1080/07370024.2017.1421954

[342] Paul Thurrott and Rafael Rivera. 2009. *Windows 7 Secrets*. John Wiley & Sons. Google-Books-ID: 1EXt2U84WZ0C.

[343] Shari Trewin, Simeon Keates, and Karyn Moffatt. 2006. Developing steady clicks: a method of cursor assistance for people with motor impairments. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility (Assets '06)*. Association for Computing Machinery, New York, NY, USA, 26–33. DOI:http://dx.doi.org/10.1145/1168987.1168993

[344] Michael L Van De Vanter. 2002. The documentary structure of source code. *Information and Software Technology* 44, 13 (Oct. 2002), 767–782.

[345] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. (Dec. 2017). DOI:http://dx.doi.org/10.48550/arXiv.1706.03762 arXiv:1706.03762 [cs].

[346] Radu-Daniel Vatavu and Ovidiu-Ciprian Ungurean. 2019. Stroke-Gesture Input for People with Motor Impairments: Empirical Results & Research Roadmap. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. DOI:http://dx.doi.org/10.1145/3290605.3300445

[347] Laton Vermette, Parmit Chilana, Michael Terry, Adam Fourney, Ben Lafreniere, and Travis Kerr. 2015. CheatSheet: A Contextual Interactive Memory Aid for Web Applications. In *Proceedings of the 41st Graphics Interface Conference (GI '15)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 241–248. http://dl.acm.org/citation.cfm?id=2788890.2788933 event-place: Halifax, Nova Scotia, Canada.

[348] Laton Vermette, Shruti Dembla, April Y. Wang, Joanna McGrenere, and Parmit K. Chilana. 2017. Social CheatSheet: An Interactive Community-Curated Information Overlay for Web Applications. *Proc. ACM Hum.-Comput. Interact.* 1, CSCW (Dec. 2017), 102:1–102:19. DOI:http://dx.doi.org/10.1145/3134737

[349] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H. Eugene Stanley, and Walter Quattrociocchi. 2016. The spreading of misinformation online. *Proceedings of the National Academy of Sciences* 113, 3 (Jan. 2016), 554–559. DOI:http://dx.doi.org/10.1073/pnas.1517441113 Publisher: National Academy of Sciences Section: Physical Sciences.

[350] Fernanda B. Viégas, Martin Wattenberg, and Kushal Dave. 2004. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. Association for Computing Machinery, Vienna, Austria, 575–582. DOI:http://dx.doi.org/10.1145/985692.985765

[351] Fernanda B. Viégas, Martin Wattenberg, and Matthew M. McKeon. 2007. The Hidden Order of Wikipedia. In *Online Communities and Social Computing (Lecture Notes in Computer Science)*, Douglas Schuler (Ed.). Springer, Berlin, Heidelberg, 445–454. DOI:http://dx.doi.org/10.1007/978-3-540-73257-0_49

[352] Chenguang Wang, Xiao Liu, and Dawn Xiaodong Song. 2020. Language Models are Open Knowledge Graphs. *ArXiv* abs/2010.11967 (2020).

[353] Ye Diana Wang and Henry H. Emurian. 2005. An overview of online trust: Concepts, elements, and implications. *Computers in Human Behavior* 21, 1 (Jan. 2005), 105–125. DOI:http://dx.doi.org/10.1016/j.chb.2003.11.008

[354] Austin R. Ward and Robert Capra. 2021. OrgBox: Supporting Cognitive and Metacognitive Activities During Exploratory Search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2570–2574. DOI:http://dx.doi.org/10.1145/3404835.3462790

[355] Sharon Watson and Kelly Hewett. 2006. A Multi-Theoretical Model of Knowledge Transfer in Organizations: Determinants of Knowledge Contribution and Knowledge Reuse*. *Journal of Management Studies* 43, 2 (2006), 141–173. DOI:http://dx.doi.org/10.1111/j.1467-6486.2006.00586.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-6486.2006.00586.x.

[356] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. (Jan. 2023). DOI:http://dx.doi.org/10.48550/arXiv.2201.11903 arXiv:2201.11903 [cs].

[357] Steve Whittaker. 2011. Personal information management: From information consumption to curation. *Annual Review of Information Science and Technology* 45, 1 (2011), 1–62. DOI:http://dx.doi.org/10.1002/aris.2011.1440450108

[358] Doug Wightman, Zi Ye, Joel Brandt, and Roel Vertegaal. 2012. SnipMatch: Using Source Code Context to Enhance Snippet Retrieval and Parameterization. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 219–228. DOI:http://dx.doi.org/10.1145/2380116.2380145 event-place: Cambridge, Massachusetts, USA.

[359] Alex C. Williams, Harmanpreet Kaur, Shamsi Iqbal, Ryen W. White, Jaime Teevan, and Adam Fourney. 2019. Mercury: Empowering Programmers' Mobile Work Practices with Microproductivity. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New York, NY, USA, 81–94. DOI:http://dx.doi.org/10.1145/3332165.3347932

[360] Parker Williams, Jeffrey Jenkins, and Joseph Valacich. 2016. Real-Time Hand Tremor Detection via Mouse Cursor Movements for Improved Human-Computer Interactions: An Exploratory Study. *SIGHCI 2016 Proceedings* (Dec. 2016). https://aisel.aisnet.org/sighci2016/10

[361] Dale M. Willows. 1974. Reading between the Lines: Selective Attention in Good and Poor Readers. *Child Development* 45, 2 (1974), 408–415. DOI:http://dx.doi.org/10.2307/1127962 Publisher: [Wiley, Society for Research in Child Development].

[362] Dale M. Willows and G. E. MacKinnon. 1973. Selective reading: Attention to the "unattended" lines. *Canadian Journal of Psychology / Revue canadienne de psychologie* 27, 3 (1973), 292–304. DOI:http://dx.doi.org/10.1037/h0082480 Place: Canada Publisher: University of Toronto Press.

[363] Max L. Wilson, Bill Kules, m. c. schraefel, and Ben Shneiderman. 2010. From Keyword Search to Exploration: Designing Future Search Interfaces for the Web. *Foundations and Trends in Web Science* 2, 1 (Jan. 2010), 1–97. DOI:http://dx.doi.org/10.1561/1800000003

[364] Heather Wiltse and Jeffrey Nichols. 2009. PlayByPlay: Collaborative Web Browsing for Desktop and Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1781–1790. DOI:http://dx.doi.org/10.1145/1518701.1518975 event-place: Boston, MA, USA.

[365] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1083–1092. DOI:http://dx.doi.org/10.1145/1518701.1518866

[366] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07)*. Association for Computing Machinery, New York, NY, USA, 159–168. DOI:http://dx.doi.org/10.1145/1294211.1294238

[367] Sungjoon Steve Won, Jing Jin, and Jason I. Hong. 2009. Contextual web history: using visual and contextual cues to improve web browser history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1457–1466. https://doi.org/10.1145/1518701.1518922

[368] Nicholas C. Wormald. 1995. Differential Equations for Random Processes and Random Graphs. *The Annals of Applied Probability* 5, 4 (1995), 1217 – 1235. DOI:http://dx.doi.org/10.1214/aoap/1177004612 Publisher: Institute of Mathematical Statistics.

[369] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yumao Lu, Zicheng Liu, and Lijuan Wang. 2022. An Empirical Study of GPT-3 for Few-Shot Knowledge-Based VQA. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 3 (June 2022), 3081–3089. DOI:http://dx.doi.org/10.1609/aaai.v36i3.20215 Number: 3.

[370] Ilan Yaniv and Shoham Choshen-Hillel. 2012. When guessing what another person would say is better than giving your own opinion: Using perspective-taking to improve advice-taking. *Journal of Experimental Social Psychology* 48, 5 (Sept. 2012), 1022–1028. DOI:http://dx.doi.org/10.1016/j.jesp.2012.03.016

[371] Taha Yasseri and Jannie Reher. 2022. Fooled by facts: quantifying anchoring bias through a large-scale experiment. *Journal of Computational Social Science* 5, 1 (May 2022), 1001–1021. DOI:http://dx.doi.org/10.1007/s42001-021-00158-0

[372] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. 2003. Faceted Metadata for Image Search and Browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. Association for Computing Machinery, New York, NY, USA, 401–408. DOI:http://dx.doi.org/10.1145/642611.642681 event-place: Ft. Lauderdale, Florida, USA.

[373] Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach. (Aug. 2019). DOI:http://dx.doi.org/10.48550/arXiv.1909.00161 arXiv:1909.00161 [cs].

[374] YoungSeok Yoon and Brad A. Myers. 2015. Supporting Selective Undo in a Code Editor. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 223–233. DOI:http://dx.doi.org/10.1109/ICSE.2015.43 ISSN: 1558-1225.

[375] Zhen Yue, Shuguang Han, and Daqing He. 2012. An investigation of search processes in collaborative exploratory web search. *Proceedings of the American Society for Information Science and Technology* 49, 1 (2012), 1–4. DOI:http://dx.doi.org/10.1002/meet.14504901386

[376] Honglei Zeng, Maher A. Alhossaini, Li Ding, Richard Fikes, and Deborah L. McGuinness. 2006a. Computing trust from revision history. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services (PST '06)*. Association for Computing Machinery, Markham, Ontario, Canada, 1. DOI:http://dx.doi.org/10.1145/1501434.1501445

[377] Honglei Zeng, Maher A. Alhossaini, Richard Fikes, and Deborah L. McGuinness. 2006b. Mining Revision History to Assess Trustworthiness of Article Fragments. In *2006 International Conference on Collaborative Computing: Networking, Applications and Worksharing*. 1–10. DOI:http://dx.doi.org/10.1109/COLCOM.2006.361890

[378] Amy X. Zhang and Justin Cranshaw. 2018. Making Sense of Group Chat Through Collaborative Tagging and Summarization. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW (Nov. 2018), 196:1–196:27. DOI:http://dx.doi.org/10.1145/3274465

[379] Amy X. Zhang, Lea Verou, and David Karger. 2017. Wikum: Bridging Discussion Forums and Wikis Using Recursive Summarization. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*. ACM, New York, NY, USA, 2082–2096. DOI:http://dx.doi.org/10.1145/2998181.2998235

[380] Jian Zhao, Michael Glueck, Petra Isenberg, Fanny Chevalier, and Azam Khan. 2018. Supporting Handoff in Asynchronous Collaborative Sensemaking Using Knowledge-Transfer Graphs. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 340–350. DOI:http://dx.doi.org/10.1109/TVCG.2017.2745279 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[381] Jingjie Zheng, Blaine Lewis, Jeff Avery, and Daniel Vogel. 2018. FingerArc and FingerChord: Supporting Novice to Expert Transitions with Guided Finger-Aware Shortcuts. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. Association for Computing Machinery, New York, NY, USA, 347–363. DOI:http://dx.doi.org/10.1145/3242587.3242589