To Search or To Gen? Exploring the Synergy between Generative AI and Web Search in Programming

Ryan Yen
University of Waterloo
r4yen@uwaterloo.ca

Nicole Sultanum
Tableau Research
nsultanum@tableau.com

Jian Zhao
University of Waterloo
jianzhao@uwaterloo.ca

ABSTRACT

The convergence of generative AI and web search is reshaping problem-solving for programmers. However, the lack of understanding regarding their interplay in the information-seeking process often leads programmers to perceive them as alternatives rather than complementary tools. To analyze this interaction and explore their synergy, we conducted an interview study with eight experienced programmers. Drawing from the results and literature, we have identified three major challenges and proposed three decision-making stages, each with its own relevant factors. Additionally, we present a comprehensive process model that captures programmers' interaction patterns. This model encompasses decision-making stages, the information-foraging loop, and cognitive activities during system interaction, offering a holistic framework to comprehend and optimize the use of these convergent tools in programming.

CCS CONCEPTS

• **Software and its engineering** → *Software design tradeoffs*; • **Information systems** → **Information retrieval**; • **Computing methodologies** → Cognitive science.

KEYWORDS

generative AI, code generation, LLM, web search, information foraging, sensemaking, cognitive science

ACM Reference Format:
Ryan Yen, Nicole Sultanum, and Jian Zhao. 2024. To Search or To Gen? Exploring the Synergy between Generative AI and Web Search in Programming. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3613905.3650867

1 INTRODUCTION

Programmers often invest time in seeking and making sense of external information to tackle programming tasks [4, 26]. Traditionally, programmers frequently engage in web searches to resolve coding challenges, such as debugging. They rely on search engines to find relevant information, error messages, and solutions shared by others in the programming community [17, 44]. However, recent advancements in Large Language Models (LLMs) have introduced an alternative information-seeking approach, consisting of generating solutions via natural language prompts. With the ability to generate customized responses for various programming scenarios, programmers turn to generative AI for tasks such as producing boilerplate code or implementing external APIs.

This raises questions about the role and coexistence of these distinct information-seeking tools in programmer workflows. Research comparing the two found that programmers often prefer generative AI over web searches when dealing with low-level code implementation due to their accessibility and adaptability [2], but continue depending on web searches to explore diverse solutions and to acquire domain-specific terms that aid them in translating their vague goals into concrete prompts for generative AI [2, 45, 52]. Another line of research seeks to integrate the functionality of these two methods by adopting retrieval-augmented generation (RAG) [23]. However, simply combining these two features contradicts programming practices, as programmers do not always accept the top search results. Instead, they rely on *signals*, such as source credibility, to assess the suitability of the results [26, 30].

While prior research has highlighted programmers' interest in combining the use of these two tools [52], the lack of a clear understanding of their intersection often leads programmers to consider them as substitutes rather than synergistic. Further, due to the inherent uncertainty and variance of both web search [4] and generative AI [32], programmers often resort to opportunistically choosing between the two tools. To address these challenges, it is essential to *understand the decisions that programmers make during the information-seeking process.* This understanding will inform future designs that consider interactions with both tools.

We conducted retrospective interviews with eight programmers experienced in both web search and generative AI for information seeking, to learn about their common practices when using both tools for programming problem-solving. Based on the results, we identified key challenges and three major decision stages, each with its own set of factors influencing the decisions. We synthesize these findings with existing literature and propose a process model by incorporating the interaction with generative AI into the information-foraging theory from Pirolli et al. [39]. This model outlines key stages of activities throughout the interaction, offering insights for future integrated designs that aim to assist programmers in effectively utilizing both web search and generative AI in their information-seeking processes.

2 RELATED WORK

To investigate the synergy between web search and generative AI, we conducted a review of programmers' information-seeking processes for both tools.

CHI EA '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0331-7/24/05.
https://doi.org/10.1145/3613905.3650867

2.1 Information Seeking and Knowledge Reusing from Web

The dynamics of utilizing web resources are multifaceted in the field of web information seeking and knowledge reuse. Programmers actively seek relevant information across various domains as described in the Information Foraging Theory (IFT) [39], such as code and search results [22, 38]. They not only gather information pertinent to their current issues [3, 15, 48] but also synthesize this information to create structured knowledge, aiding in their decision-making [18, 19, 27]. A prevailing challenge is comprehending the rationale behind previous programmers' decisions due to inadequate or outdated documentation [16, 46], which underscores the importance of effective knowledge documentation and retrieval. In this context, knowledge reuse becomes significant, as it entails not just generating new knowledge but applying existing knowledge to problem-solving [10, 14, 24, 25, 53]. Building upon this previous literature, our study focuses on the intersection between information-seeking via the web and generative AI, particularly in terms of understanding, translating, and reusing knowledge from one tool to another.

2.2 Generative AI-assisted Problem-Solving

Similar to web searches, using generative AI for programming problem-solving also involves information-seeking and sense-making processes, yet with distinct workflows and challenges. Programmers using generative AI in their problem-solving often encounter limitations within the linear question-answer paradigm, which hinder exploration and the ability to revisit previous responses [2, 43, 54]. LLMs are also capable of generating plausible results for NL prompts that may not necessarily be aligned with the current usage scenario, which exacerbates friction [13, 28, 56, 58]. For instance, writing *"Scrape this web page with JavaScript"* can already generate a program without syntax error, suggesting the use of Node.js for web scraping. Nevertheless, programmers might also need to visualize the scraped data on a self-hosted website, which necessitates web scraping with a server-side rendering front-end JavaScript framework. As a result, programmers might find themselves ensnared in a debugging rabbit hole [32, 58], devoid of the guidance that web searches offer in terms of evaluating the credibility and suitability of particular answers [52].

Prior work has also incorporated information retrieval techniques into the generation process of LLMs [23], offering access to real-world and up-to-date information. Our work believes programmers' active involvement in the information-seeking and sensemaking process is essential for them to iterate on prompts or search queries in their pursuit of the most suitable results. Therefore, further investigation is warranted to explore the intersection, challenges, and requirements when interacting with both web search and generative AI.

2.3 AI-assisted Decision Making

AI-assisted or human-AI decision-making has been explored in various contexts with a broader goal of evaluating, understanding, and enhancing the human experience and performance in decision-making tasks [20]. These studies have examined different decision-making tasks, various techniques to improve performance [21, 29, 31], and diverse evaluation metrics to assess both tasks and AI [41, 47]. In investigating the decision-making stages of programmers when interacting with retrieval models and language models, we discerned that various factors influencing programmers' decisions coincided with existing research in the field, such as familiarity with the domain [8]. Our work aims to be the first exploration of programmers' decision-making stages during information-seeking using both tools. We anticipate that future research will explore deeper into evaluating comprehensive metrics and strategies to enhance programmers' decision-making in these stages.

3 INTERVIEW STUDY

To gain insights into the practices employed by programmers, challenges and factors in their decision-making, we conducted retrospective interviews with experienced programmers.

3.1 Participants and Procedure

We recruited eight participants (5 males, 3 female; ages $24-29$, $M = 26.8, SD = 1.26$) through purposive sampling [11]. In our recruitment process, we sought participants experienced in programming and using LLM-driven code generation tools. Eligibility screening involved a pre-test survey that assessed participants' self-reported programming experience on a 5-point scale [1: very inexperienced; 5: very experienced], years of programming experience, and self-reported familiarity with LLM-driven code generation tools (Pre-test survey in Appendix A). All recruited participants reported having more than four years of programming experience ($M = 5.43$ years, $SD = 1.12$) and were confident in their programming experiences (score $M = 4.29, SD = 1.08$), familiar with LLM-code generation tools (score $M = 4.41, SD = 0.37$), and regularly used the LLM-code generation tools ($M = 12$ times/week, $SD = 4.19$).

Participants were compensated with 20 CAD for a 45-minute interview session. We first asked each participant to provide a minimum of three recent examples of their ChatGPT usage for programming problem-solving, including instances involving web search as part of the process, to encourage participants to reflect on their utilization of both web search and generative AI. We then asked about challenges they faced when interacting with both tools, explored scenarios involving the combined usage, and inquired about their thought processes throughout the information-seeking process (Interview Questions in Appendix A.1).

All interviews were audio-recorded and subsequently automatically transcribed [1] into written text. We analyzed the interviews using thematic analysis [6], employing both inductive and deductive approaches. After conducting interviews with eight participants, two researchers in the team independently conducted the initial analysis in theme identification related to challenges, decision-making stages, influential factors, and knowledge extraction. The agreement on factors of the three decision-making stages was quantified using inter-rater reliability, reaching 84%. For knowledge extraction, the inter-rater reliability was calculated at 92%, using Cohen's kappa. After this initial coding, a collaborative session involving the research team ensued, employing an open coding approach [9] for further refinement and categorization. Three iterations of discussions with the research team were conducted to address any discrepancies in coding and theme categorization. The collaborative efforts culminated in achieving 100% agreement on the identified themes.

3.2 Collected Data

All participants provided at least three example scenarios in which they employed both web search and generative AI for information-seeking and problem-solving. In the majority of scenarios (26 out of 28), participants engaged in more than two rounds of iterations involving both web search and prompting, with two cases involving only one web search session and one round of prompting.

Regarding the tasks participants undertook, we identified 21 out of 28 of them as open-ended tasks where participants did not require specific approaches to solve them. Among these, 9 pertained to exploratory data analysis and modelling tasks, 5 involved front-end development, 4 related to data mining and web scraping, and the remaining 3 concerned server-related tasks.

4 CHALLENGES

Results from the interview study indicate that programmers require assistance in making informed decisions about which tool to use (C1), determining which results to extract and apply (C2), and translating results into concrete search queries or prompts for subsequent iterations (C3).

4.1 C1 - Lack of Guidance to Determine Tool Selection and Integration

In the majority of scenarios presented by participants, either web search or generative AI was seen as a fallback to the other when each failed. For example, P6 mentioned, *"I would try using web search when ChatGPT kept giving me the wrong answer."* There were also several scenarios where combining both tools produced the most favourable outcomes, as *"they each possessed their strengths."* -P2. Additionally, participants did not report explicit metrics to determine which tool should be used next, stating they *"cannot anticipate how the results will appear"* -P1 or *"what knowledge I [they] will gain"* -P3 in the current round. The choice of what approach to try next still relies on trial and error. Therefore, it is relevant to comprehend the factors that influence the choice and offer programmers guidance on which tool to use in various circumstances.

4.2 C2 - No Scaffolding to Extract Information from Results

Most participants have developed their own strategies for understanding and extracting information gained from either generated results or web search results. However, the majority of participants (7 out of 8) *did not mention specific metrics to evaluate the appropriateness of the results.* For instance, P2 stated that the appropriateness of generated results was mostly based on *"experience"* and *"when the results are similar to what I [they] expected"*. In most scenarios, participants were unable to explicitly express the sources of the knowledge they extracted from. For instance, P4 mentioned, *"I did write a new prompt based on the knowledge I gained from web pages I visited, but I could not tell exactly where it is from."* This highlights the need for provenance tracking for externalizing programmers' information-seeking process, consistent with prior research [37].

4.3 C3 - Difficulties in Adapting Results between Search and Generative AI

All participants reported difficulties in switching between the two tools due to the lack of understanding regarding *"the exact differences"* -P5 between them. Participants often struggled with translating results from one format to another to derive *"the best outcome from both inputs."* -P5 Some (3 out of 8) also explicitly expressed a desire for a more seamless integration of both tools. P4 suggested that integrating web search windows into the code editor could help them *"focus on iterating the code generation,"* viewing web search as a *"supplementary means to enhance generation."* -P7 This suggests a shared interest among participants in a more streamlined workflow for efficient generation with external resources.

5 DECISION-MAKING STAGES

To summarize the activities of programmers during the information-seeking process, we outline three decision stages that programmers encounter, each associated with the challenges mentioned above (see Fig. 1):

1. The *Selection* stage, during which programmers choose between the two available tools (C1).
2. In the *Extraction* stage, programmers assess whether to utilize implicit or explicit knowledge (C2).
3. During the *Translation* stage, programmers transform their internal knowledge into natural language queries or prompts (C3).

5.1 Selection Stage

We identified seven major factors that influence programmers' decisions on *which tools to use in each iteration.*

5.1.1 *Familiarity with the Domain.*

Participants use web searches to familiarize themselves with how to construct their prompts, particularly when they are not familiar with the domain. In such scenarios, participants understand that the results generated by AI may not provide the answer, and they *"fear that I [they] do not possess the knowledge to verify correctness."* -P1. Conversely, participants tended to employ generative AI when they were more *familiar* with the task at hand, such as using APIs they may have forgotten or implementing detailed algorithms.

5.1.2 *Clarity of Goals.*

Participants generally turned to web searches when their goals were unclear and less defined. In such situations, they struggle to *"determine which keywords are appropriate"* -P6 and which approaches to take. The diversity of web search results offers them a better understanding of potential solutions. Once the problem becomes more well-defined, participants prefer to use generative AI, as their primary objective at that point is to *"find the most aligned [generated] results."* -P1. Another condition that leads participants to opt for generative AI over web search is when they believe that the problem is well-defined and has been thoroughly discussed in the past. P5 explained, *"I believe this problem has enough solutions being trained as data to the [AI] model."*

5.1.3 *Repetition of Results.*

Participants often find themselves trapped in a debugging cycle where they continuously iterate on the prompt and repetitively receive incorrect results. Previous research has reported this issue when users repeatedly receive results that do not

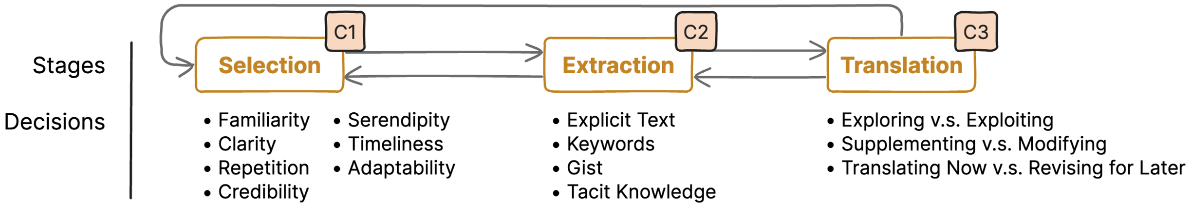

Figure 1: Three decision-making stages with key factors influencing these decisions and the challenges associated with them.

align with their intentions, causing them to struggle to find the right terms to prompt the LLM. The majority of participants (6 out of 8) mentioned that they would fallback to web searches when they realized they were *"stuck in the loop."* -P2 P6 elaborated, *"I will still try a few times if the [AI-generated] results do not match before turning to the web search."*

5.1.4 Credibility and Diversity. Participants believe that by glancing through the appropriateness of multiple search results, they can easily *"identify the keywords to use for prompts"* -P8 and determine the *"overall approach they should take"* -P3 to solve the problem, which echos prior research [12, 30]. Four participants mentioned that they would use web searches to validate the correctness of the generated results before applying them. They suggested that web searches provide *"more detailed explanations from different aspects"* -P3 and provide additional signals for assessing credibility, such as upvotes on StackOverflow.

5.1.5 Serendipity and Luck. While we observed that most circumstances begin with a web search when the problem is vague, we also noticed several scenarios where participants opt for generative AI in the hope that it might opportunistically provide the final result. This finding echoes previous research that uncovered similar behaviours [33, 58], similar to opportunistic programming [5].

5.1.6 Up-to-dateness of Information. The timeliness of information is usually considered a crucial aspect of resource credibility [26, 30, 51]. Participants reported that they tend to opt for web searches when they require the most up-to-date results, particularly when they seek the latest documentation for packages or libraries. However, participants pointed out that the overall approach they follow does not necessarily have to be up to date; instead, only the low-level code implementation requires currency. For instance, when P6 was working on creating a responsive design, the high-level strategy remained consistent, but the low-level code required updates to align with current browser capabilities (e.g., specific CSS properties for responsiveness).

5.1.7 Customizability and Adaptability. When adapting solutions back into their programs, all participants relied on generative AI. They preferred this approach because of the customizability and adaptability of the generated results. P5 explained that *"GPT kind of combines solutions for you,"* and P7 mentioned that *"I do not have to change the parameters or variables."* However, two participants voiced concerns about overconfidence, as they occasionally bypassed the validation phases and overlooked inaccuracies.

5.2 Information Extraction Stage

At this stage, programmers must discern *which results are useful for the next iteration.* We identified two relevant factors.

5.2.1 Role of Early Iterations. All participants reported that the results from the first few iterations are not worth reading in detail, especially when the problem is vague or relatively complex. The primary goal of these initial iterations is to *"narrow down the scope [of the solution]"* -P6 and *"define the problem domain."* -P1. Most participants (7 out of 8) also mentioned that these initial rounds serve to understand whether the search query or prompt can provide results in the *"right direction."* -P7. Thus, participants need to quickly skim through either the search results or the generated content to identify any misalignment with their intentions. For example, P2 asked the generative AI to visualize a dataset in a bubble chart with a prompt like *"Scatter plot with some dots are larger [...]."* The results showed a scatter plot where the marker size changed. P2 skimmed through the visualization part in the step-by-step tutorial and realized that implementing a scatter plot might be too complex. Participants then had to decide whether to switch to another tool or continue iterating to obtain a more aligned solution.

5.2.2 Extract Text, Keywords, Gist, or Tacit Knowledge. We identified four major elements that programmers reused and carried forward to the next iteration through iterative open coding. At the most concrete level, participants often directly copy-pasted extracted ●*text* from results either into the search query or as the context of the prompt. This was especially common when search results were lengthy, and participants preferred not to organize them themselves. When iterating on the prompt or search query itself, most participants (6 out of 8) derived ●*keywords* from previous iterations and used them to guide the direction of the next solutions. In certain instances, participants engaged in the translation of the gathered insights, represented as the ●*gist*, which amalgamated information from multiple solutions. These *gists* primarily encapsulate the general directions indicated by the current prompt or query. The importance of these curated *gists* lies in their role in evaluating the alignment of participants' intentions with the models. These insights are forwarded to the next round without direct replication of specific segments from the results. This approach was evidenced when participants started another round right after scrolling through search results without clicking on any pages. In the most abstract levels, participants applied implicit ●*knowledge* gained from the results. These knowledge refer to what programmers have learned or derived from the foraging process, including insights into effective problem-solving strategies, the identification

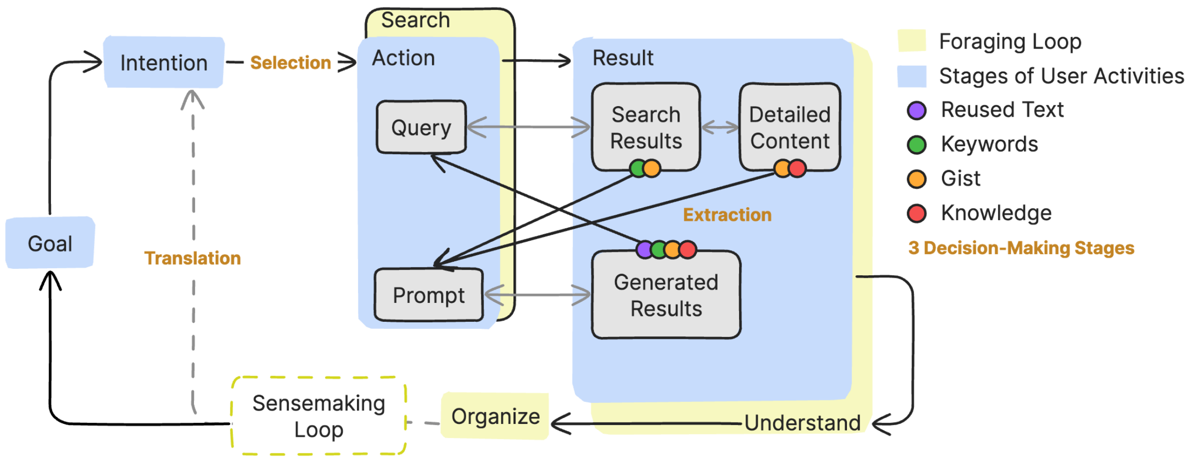

Figure 2: A process model describing interactions between programmers and search/generative AI.

of optimal coding practices, a deeper understanding of system architectures, and the acquisition of domain-specific knowledge crucial for informed decision-making in software development. However, they did not always apply it directly to the next round; instead, it was sometimes more *"useful when verifying the next results."* -P8.

5.3 Knowledge Translation Stage

In the final stage of each iteration, programmers must decide *how to translate their knowledge into text*, whether it be a search query or a prompt for generative AI. We have identified three quandaries that programmers should weigh as they trade off various considerations.

5.3.1 Exploring vs. Exploiting. The most common dilemma is the decision of whether to persist with iterations on the current topic or explore a wider range of topics. This finding aligns with the exploration and acceleration modes discovered by Barke et al [2]. Participants either rewrote the entire prompt or query to ask different questions or refined the existing prompt or query to dive deeper into the same domain. The challenge arises when programmers are uncertain *"if the information is sufficient"* -P7 for either generative AI or crafting web search queries. Programmers might proceed to the next exploration without fully grasping the solution. Consequently, they encounter difficulties in tracing back to search or prompt histories without analytical provenance [34, 36, 55].

5.3.2 Supplementing vs. Modifying. Participants have the option to either add results into the prompt or query as context or directly modify the original prompt or query. The former is more common when programmers have clearer intentions in mind, while the latter is employed when they are still in the phase of adjusting the direction of information seeking. For example, P4 sought information about implementing an API in the Next.js framework using generative AI. Initially, P4 tended to modify the prompt to understand where to implement the code and the differences between code versions. After deciding on approaches, P4 then pasted all the documentation and tutorials as context for AI to generate results.

5.3.3 Translating Now vs. Reserving for Later. Similar to previous research [32], which suggests that programmers often set aside some generated results for later use, our findings indicate that several participants (5 out of 8) may not immediately apply the knowledge acquired in one round to the subsequent round. Rather than a *linear chain* of knowledge, this process resembles a more intricate *tree diagram*, with various branches and connections as participants navigate and adapt their strategies. Participants can revisit and apply acquired knowledge later after exploring different branches. For instance, they might explore one branch, then switch to another, and eventually, go back to the first branch to combine insights for a more comprehensive solution.

6 TOWARDS A PROCESS MODEL FOR PROGRAMMER-SEARCH/GENERATIVE AI INTERACTION

When interacting with both web search and generative AI, programmers face challenges at each decision stage, particularly in selecting the tool to use, determining which information to extract, and translating information into concrete text. Prior research has described the stages of information foraging [40, 42, 50], offered frameworks for determining information appropriateness [17, 26], and explored cognitive models for translating vague goals into natural language prompts in AI-driven systems [35, 49, 57]. However, without a clear understanding of the synergy between these two tools, current designs often focus on one or the other individually instead of considering them as a whole. While web search has been integrated further into generative AI [23], current tools have not fully taken into account the interaction among information foraging loop and the value of the human cognitive thought process.

To help guide future design, we summarize our findings in the form of a *process model* that enables parallel interactions with web search and generative AI. This model unifies our insights and those of prior studies, including principles of the sensemaking foraging loop [40] and Norman's seven stages of activities [35] (see Fig. 2) to depict decision-making iterations. The flow begins with a **Goal**, where programmers identify their problems and tasks. In the pursuit of goals, programmers then articulate **Intentions** for addressing them. For instance, when aiming to classify the Iris dataset, programmers dissect the problem into selecting an appropriate model and outlining the training and evaluation procedures. They then either leverage web searches or generative AI systems as **Actions**. We termed this decision point as the *Selection* stage, which involves choosing the most suitable tool for the given context.

Upon executing their actions, programmers receive a list of search results or AI-generated content. Akin to the foraging loop's **Understanding** step, they analyze and interpret these results, identifying essential information for *Extraction*. Continuing with our example, a programmer might encounter various model suggestions in their search results, select the keyword *"logistic regression"*, and delve deeper into this specific approach. The *Extraction* stage involves distilling information from the results, encompassing explicit ●*text*, relevant ●*keywords*, the ●*gist* from the overall results, and abstract ●*knowledge* gained during the process.

Programmers then **Organize** this acquired knowledge, preparing it for the *Translation* stage. This final decision point involves reformulating the information into a new search query or AI prompt. For example, they might integrate steps from a web tutorial with code examples to guide generative AI. These activities among the **Gulf of Evaluation** are essentials to determine the next step in the iteration. It can lead back to the **Goal** stage, particularly when the initial objective is broad and requires refinement, or return to the **Intention** stage, where programmers refine their mental models for problem-solving. This cyclical process dynamically adapts to the evolving needs and understanding of the programmer, facilitating effective use of both web search and generative AI tools.

Incorporating the Sensemaking Loop. While this study did not directly investigate the **sensemaking** process, it is noteworthy that three participants mentioned the importance of note-taking to organize their collected information. This finding aligns with prior research indicating that programmers engage in a sensemaking loop before embarking on the next iteration of information foraging [34, 40, 42, 50]. Incorporating this sensemaking process into the process model typically occurs during the evaluation process, starting from the **results**. Programmers articulate the knowledge they have internalized before making adjustments to their **intentions** or **goal**. Future research could explore how programmers externalize their curated knowledge to generate results with context. Additionally, investigating how sensemaking influences the proposed decision stages, with the potential to enhance the translation stage, would be valuable.

Implications for Reconciling both Tools. Our study aims to serve as an exploratory investigation to shed light on the challenges, practices, and decision-making processes across the stages of information-seeking. We decompose the information-seeking process into three main decision-making stages. Future design endeavours can delve deeper into these three stages, examining the detailed needs required to assist programmers in making informed decisions throughout the information-seeking process. This would be particularly beneficial in terms of enhancing the controllability of the process, enabling programmers to steer the system toward their goals. Overall, the insights gleaned from this study can inform future designs by advocating for the incorporation of the programmer-foraging process into the design of retrieval augmented generation.

7 LIMITATIONS AND FUTURE WORK

While valuable insights were gained from the interviews about programmers' decision-making during web information foraging, we note the limitations of this method. The decision to opt for retrospective interviews aimed at gaining in-depth insights into programmers' experiences during information foraging stages, enabling a qualitative exploration of cognitive processes [7]. Thus, we encouraged participants to share multiple cases, enhancing our understanding of diverse web foraging scenarios and providing a comprehensive perspective on the dynamic nature of these processes. Acknowledging the limitations of interviews in capturing micro-decisions prone to errors and biases, future work could integrate an observation study with think-aloud protocols or activity logging in a diary study for a more quantitative analysis.

Furthermore, while our study provides insights into how programmers can actively participate in the retrieval process as information retrievers, we also recognize the limitations of this approach compared to current human-in-the-loop retrieval augmentation generation methods and their potential necessity in the future. A notable difference is that in those approaches, humans are involved by annotating the usefulness of the retrieval results, whereas our approach involves humans directly in the entire retrieval process throughout different stages.

8 CONCLUSION

In this paper, we investigated the intersection of web search and generative AI in programming, uncovering key challenges and decision-making stages. Our findings from interviews with experienced programmers reveal a nuanced relationship between these tools, highlighting the need for a synergistic approach rather than treating them as alternatives. We then propose a process model that integrates web search and generative AI into programming workflows, emphasizing the importance of understanding, extracting, and translating information across tools.

ACKNOWLEDGMENTS

This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) via the Discovery Grant and the University of Waterloo via the Faculty Start-up Fund.

REFERENCES

[1] 2023. Otter.AI -AI Meeting Note Taker and Real-time AI Transcription. https://otter.ai/

[2] Shraddha Barke, Michael B James, and Nadia Polikarpova. 2023. Grounded copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages* 7, OOPSLA1 (2023), 85–111.

[3] 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*. 513–522.

[4] Joel Brandt, 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*. 1589–1598.

[5] Joel Brandt, Philip J Guo, Joel Lewenstein, and Scott R Klemmer. 2008. Opportunistic programming: How rapid ideation and prototyping occur in practice. In

Proceedings of the 4th international workshop on End-user software engineering. 1–5.

[6] Virginia Braun and Victoria Clarke. 2012. *Thematic analysis.* American Psychological Association.

[7] Gabriele Budach. 2012. Part of the puzzle: the retrospective interview as reflexive practice in ethnographic collaborative research. In *Multilingualism, Discourse and Ethnography*, Sheena Gardner and Marilyn Martin-Jones (Eds.). Routledge, 319–333. https://eprints.soton.ac.uk/345587/

[8] Carrie J Cai, Emily Reif, Narayan Hegde, Jason Hipp, Been Kim, Daniel Smilkov, Martin Wattenberg, Fernanda Viegas, Greg S Corrado, Martin C Stumpe, et al. 2019. Human-centered tools for coping with imperfect algorithms during medical decision-making. In *Proceedings of the 2019 chi conference on human factors in computing systems.* 1–14.

[9] Kathy Charmaz. 2006. *Constructing grounded theory: A practical guide through qualitative analysis.* sage.

[10] Thomas Davenport, Sirkka Jarvenpaa, and Michael Beers. 1996. Improving knowledge work processes. *MIT Sloan Management Review* (1996).

[11] Ilker Etikan, Sulaiman Abubakar Musa, Rukayya Sunusi Alkassim, et al. 2016. Comparison of convenience sampling and purposive sampling. *American journal of theoretical and applied statistics* 5, 1 (2016), 1–4.

[12] 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* 324, 7337 (2002), 573–577.

[13] Felicia Li Feng, Ryan Yen, Yuzhe You, Mingming Fan, Jian Zhao, and Zhicong Lu. 2023. CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural Language Programming. arXiv:2310.09235 [cs.HC]

[14] 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.* 1–12.

[15] Raphael Hoffmann, James Fogarty, and Daniel S Weld. 2007. Assieme: finding and leveraging implicit references in a web search interface for programmers. In *Proceedings of the 20th annual ACM symposium on User interface software and technology.* 13–22.

[16] Amber Horvath, Brad Myers, Andrew Macvean, and Imtiaz Rahman. 2022. Using Annotations for Sensemaking About Code. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology.* 1–16.

[17] 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).* IEEE, 305–306.

[18] 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.* 999–1010.

[19] 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.* 2989–2998.

[20] Vivian Lai, Chacha Chen, Alison Smith-Renner, Q. Vera Liao, and Chenhao Tan. 2023. Towards a Science of Human-AI Decision Making: An Overview of Design Space in Empirical Human-Subject Studies. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23).* Association for Computing Machinery, New York, NY, USA, 1369–1385. https://doi.org/10.1145/3593013.3594087

[21] Vivian Lai and Chenhao Tan. 2019. On Human Predictions with Explanations and Predictions of Machine Learning Models: A Case Study on Deception Detection. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19).* Association for Computing Machinery, New York, NY, USA, 29–38. https://doi.org/10.1145/3287560.3287590

[22] Joseph Lawrance, Rachel Bellamy, and Margaret Burnett. 2007. Scents in programs: Does information foraging theory apply to program maintenance?. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007).* IEEE, 15–22.

[23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[24] 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.* 577–589.

[25] 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.* 67–80.

[26] 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 (2021), 1–35.

[27] 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.* 1–16.

[28] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D Gordon. 2023. "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.* 1–31.

[29] Shuai Ma, Ying Lei, Xinru Wang, Chengbo Zheng, Chuhan Shi, Ming Yin, and Xiaojuan Ma. 2023. Who Should I Trust: AI or Myself? Leveraging Human and AI Correctness Likelihood to Promote Appropriate Trust in AI-Assisted Decision-Making. https://doi.org/10.48550/arXiv.2301.05809 arXiv:2301.05809 [cs].

[30] 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.

[31] Katelyn Morrison, Donghoon Shin, Kenneth Holstein, and Adam Perer. 2023. Evaluating the Impact of Human Explanation Strategies on Human-AI Visual Decision-Making. *Proceedings of the ACM on Human-Computer Interaction* 7, CSCW1 (April 2023), 48:1–48:37. https://doi.org/10.1145/3579481

[32] Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. 2022. Reading between the lines: Modeling user behavior and costs in AI-assisted programming. *arXiv preprint arXiv:2210.14306* (2022).

[33] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2023. In-IDE Generation-based Information Support with a Large Language Model. *arXiv preprint arXiv:2307.08177* (2023).

[34] Phong H Nguyen, Kai Xu, Andy Bardill, Betul Salman, Kate Herd, and BL William Wong. 2016. SenseMap: Supporting browser-based online sensemaking through analytic provenance. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST).* IEEE, 91–100.

[35] Donald A Norman. 1986. Cognitive engineering. *User centered system design* 31, 61 (1986), 2.

[36] Chris North, Remco Chang, Alex Endert, Wenwen Dou, Richard May, Bill Pike, and Glenn Fink. 2011. Analytic provenance: process+ interaction+ insight. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems.* 33–36.

[37] 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.* 1–14.

[38] Alexandre Perez and Rui Abreu. 2014. A diagnosis-based approach to software comprehension. In *Proceedings of the 22nd International Conference on Program Comprehension.* 37–47.

[39] Peter Pirolli and Stuart Card. 1999. Information foraging. *Psychological review* 106, 4 (1999), 643.

[40] 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*, Vol. 5. McLean, VA, USA, 2–4.

[41] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Wortman Vaughan, and Hanna Wallach. 2021. Manipulating and Measuring Model Interpretability. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21).* Association for Computing Machinery, New York, NY, USA, 1–52. https://doi.org/10.1145/3411764.3445315

[42] 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.* 608–618.

[43] Steven I Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D Weisz. 2023. The programmer's assistant: Conversational interaction with a large language model for software development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces.* 491–514.

[44] 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.* 191–201.

[45] Advait Sarkar, Andrew D Gordon, Carina Negreanu, Christian Poelitz, Sruti Srinivasa Ragavan, and Ben Zorn. 2022. What is it like to program with artificial intelligence? *arXiv preprint arXiv:2208.06213* (2022).

[46] 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.* 23–34.

[47] Fabian Sperrle, Mennatallah El-Assady, Grace Guo, Duen Horng Chau, Alex Endert, and Daniel Keim. 2020. Should We Trust (X)AI? Design Dimensions for Structured Experimental Evaluations. https://doi.org/10.48550/arXiv.2009.06433 arXiv:2009.06433 [cs].

[48] Jeffrey 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). IEEE, 195–202.

[49] Hariharan Subramonyam, Christopher Lawrence Pondoc, Colleen Seifert, Maneesh Agrawala, and Roy Pea. 2023. Bridging the Gulf of Envisioning: Cognitive Design Challenges in LLM Interfaces. *arXiv preprint arXiv:2309.14459* (2023).

[50] Sangho Suh, Bryan Min, Srishti Palani, and Haijun Xia. 2023. Sensecape: Enabling Multilevel Exploration and Sensemaking with Large Language Models. *arXiv preprint arXiv:2305.11483* (2023).

[51] Marsha Ann Tate. 2009. *Web wisdom: How to evaluate and create information quality on the Web.* CRC Press.

[52] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts.* 1–7.

[53] 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.* 241–248.

[54] Frank F Xu, Bogdan Vasilescu, and Graham Neubig. 2022. In-ide code generation from natural language: Promise and challenges. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 2 (2022), 1–47.

[55] Kai Xu, Simon Attfield, TJ Jankun-Kelly, Ashley Wheat, Phong H Nguyen, and Nallini Selvaraj. 2015. Analytic provenance for sensemaking: A research agenda. *IEEE computer graphics and applications* 35, 3 (2015), 56–64.

[56] Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, and Li Yuan. 2023. Llm lies: Hallucinations are not bugs, but features as adversarial examples. *arXiv preprint arXiv:2310.01469* (2023).

[57] Ryan Yen, Jiawen Zhu, Sangho Suh, Haijun Xia, and Jian Zhao. 2023. CoLadder: Supporting Programmers with Hierarchical Code Generation in Multi-Level Abstraction. *arXiv preprint arXiv:2310.08699* (2023).

[58] JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems.* 1–21.

A SURVEY AND INTERVIEW QUESTIONS

ELIGIBILITY SCREENING SURVEY

1. **How confident are you in your overall programming experience?**
   1: Very Inexperienced
   2: Inexperienced
   3: Moderately Experienced
   4: Experienced
   5: Very Experienced
2. **How many years of programming experience do you have?** years
3. **How familiar are you with AI code generation tools (e.g., GitHub Copilot, ChatGPT)?**
   1: Not Familiar
   2: Slightly Familiar
   3: Moderately Familiar
   4: Familiar
   5: Very Familiar
4. **Over the past few weeks, how often did you typically employ AI code generation tools such as OpenAI's Codex, GitHub Copilot, or ChatGPT for your programming tasks?** (e.g., times per week)

A.1 Semi-Structured Interview Questions

Programming Workflow and Tool Integration

1. **Programming Workflow Integration:** Can you describe your typical programming workflow, particularly emphasizing how you utilize code synthesis tools like Copilot and web search in this process?
2. **Decision Making between GPT and Web Search:** How do you decide when to use tools like GPT and when to resort to web search during your coding process? Could you provide a specific example illustrating this decision-making process?

Information Seeking and Evaluation in Programming

1. **Information Requirements:** When you begin looking for information during programming, what specific types of information are you usually seeking? (e.g., syntax clarification, algorithmic approaches, best practices)
2. **Assessment of Information Quality:** What criteria do you use to determine whether a search or generated result is good enough for your needs? What factors are important to you?
3. **Determining Importance of Information:** What kind of information do you consider as most important from the generated or search results?
4. **Synthesizing Information from Multiple Sources:** Can you describe how you synthesize or combine information from different sources (like Copilot, web search, forums)? How do you resolve conflicts or discrepancies in information?
5. **Long-term Information Retention:** When you find particularly valuable information, how do you ensure its retention for future use? Do you have a system for organizing or bookmarking useful resources?

Challenges and Limitations of Both Tools

6. **Challenges and Limitations:** Could you discuss some challenges or limitations you've encountered with both tools? How does the other tool help in overcoming these challenges?
7. **Web Search Efficacy:** Can you provide an example where web search helped you gain a better understanding of a programming concept or language feature that tools like Copilot alone couldn't provide?
8. **Future of GPT and Web Search:** In your opinion, do you think the advancement of technologies like GPT-4 with internet and web scraping access could eventually replace traditional web search for programming-related queries?