

A Text Foraging Perspective on Processing Code

Teresa Dreyer

PLTlab.de

Marburg University

tdreyer@mathematik.uni-marburg.de

Jan Tünnermann

Charlotte Fresenius University

of Psychology

jan.tuennermann@charlotte-fresenius-uni.de

Christoph Bockisch

PLTlab.de

Marburg University

bockisch@acm.org

Abstract

This paper outlines research aimed at investigating similarities and differences between human search patterns in ordinary texts and in code through the lens of the foraging paradigm. Visual foraging studies transfer methods from behavioral ecology to the study of human search. Information foraging theory applies these methods to studying human interaction with information technology. The perspective of information foraging in code navigation and comprehension was explored in previous research. Because code is a special form of text, it should be expanded by the text dimension. Due to its highly formalized structure with non-linear control flow, code resembles a natural foraging scenario more than regular text. We are therefore interested in how human search patterns in code compare to those found for normal text. We outline an experimental approach to that end.

1. Introduction

When programmers interact with code, they spend a substantial amount of time with reading and searching through code; their own and other developers' code as well as code generated by generative AI or coding agents. This includes simple searches, such as searching for a specific term by manually scrolling through the code or by skipping through candidates using the IDE's "find next" feature. However, many search tasks in program code are more complex: The implementation of one concern is often distributed over multiple program locations or files. As a consequence, tasks focused on gaining an overview and understanding of the code, such as code navigation, program comprehension, debugging, or code audits require complex searches, including multiple search targets and jumping between different code locations or files. Such tasks have in common that there is often uncertainty regarding whether and how many of the desired pieces of information might be present at the current location.

Three observations primarily motivate our work:

Observation 1: The complex code search scenario with multiple uncertain targets in distributed environments resembles the challenge animals face when foraging for food in a patchy environment, a research topic addressed in behavioral ecology (Charnov, 1976).

Observation 2: Because code is a special form of text, insights from text processing can be transferred to understanding how programmers process code. But program code has special structural features not found in common text, which may pose additional challenges and opportunities.

Observation 3: The presentation of program code is often visually enhanced. Visual enhancement can benefit from knowledge about human visual perception and attention guidance, topics with a long tradition in psychological research.

Visual foraging has been extensively studied and connections to text and code processing have been investigated. However, a comprehensive synthesis of both perspectives that also takes into account

visual enhancements aimed at understanding how programmers navigate and search through source code – and at identifying ways to better support them in this process – remains an open research challenge. The present paper provides a brief overview of the related literature and proposes an empirical research agenda to systematically investigate these interactions and inform the design of more effective developer support tools.

2. State of the Art and Problem Statement

Human visual search has been extensively studied in cognitive psychology (Wolfe, 2020). Research in this domain often focuses on finding a target object (often abstract shapes) among several “distractors” and investigates the impact of the stimuli’s low-level visual properties and the searchers’ top-down knowledge. This paradigm could be considered a model for the simple code searches mentioned above. When turning to the more complex search tasks associated with program comprehension, visual foraging seems to be a more appropriate model. This can be illustrated when mapping code search to an (animal) food foraging example: A code environment (a file, a module, or just some chunk of code) resembles a berry bush. It contains multiple relevant pieces of information, the berries, but also a lot of clutter, the leaves and other greenery. The clutter consists of code and comments that are not relevant to the current search task. The berry foragers (programmers) are motivated to search efficiently in order to fill their bellies quickly (complete the complex search task).

The foraging analogy has been applied to multi-target visual search, where human participants have to search for multiple instances of simple visual targets (such as shapes) among distractors. This allows not only to investigate search efficiency with regard to visual target and distractor features, but also a decisional component: when to leave one patch (search environment) to continue searching in the next environment (Wolfe, 2013; Grössle, Schubö, & Tünnermann, 2023). The foraging paradigm has also been applied to search in common text. Within the area of text processing research, text skimming, the act of quickly reading a text to get an overview over its contents, is of particular interest to us because it resembles quickly looking through an unfamiliar code project to gain a general understanding. In the text skimming context, foraging has been shown to be a suitable approach: Research by Duggan and Payne (2009) has investigated text skimming as a technique of foraging for relevant information in text. Their results indicate that readers are effective at foraging for relevant information in text when under time pressure and suggest that readers aim to maximize useful information while minimizing time and search effort. Chi, Gumbrecht, and Hong (2007) took the perspective that text processing in general moves in the direction of skimming and found tentative evidence that visual markups such as highlights are effective at directing readers’ attention to relevant text sections. This suggests that such markups improve readers’ foraging for information.

A more general perspective is provided by Information foraging theory, which applies a foraging perspective to humans’ search for information (Pirolli & Card, 1995, 1999). Such research has demonstrated that effective information foraging depends on strong, meaningful cues (“information scent”) that help users quickly predict where valuable information is located. Information foraging theory has inspired research in several code-processing related domains, among others code navigation and debugging. The approach has been shown to be suitable for such tasks: Research by Niu, Mahmoud, and Bradshaw (2011) in the context of code navigation graphs indicates that foraging principles are applicable to code navigation which is one of our code comprehension scenarios. In the area of another of our code comprehension scenarios, debugging, Lawrance et al. (2013) presented an information-foraging-based theory on how programmers navigate code during debugging. Based on information scents, they introduced a model for programmers’ code navigation behavior in a debugging task which they found to be a good predictor, further strengthening our proposed approach.

The approaches discussed above have in common that optimality can often be quantified using optimal foraging theory (OFT; Charnov, 1976) a framework from behavioral ecology to model animal food foraging. A central prediction of OFT is that foragers optimize their behavior according to marginal value theorem (MVT) by leaving a patch when the instantaneous yield from that patch drops below the

average yield experienced in the overall environment.

We are interested in *comparing how programmers search for information in code versus how humans search for information in common text, with the goal to develop better strategies for gaining an overview and understanding of code*. In particular, we expect MVT-based predictions for code search to outperform those for text search. The rationale is that the code searching scenario more closely resembles natural food search: Compared to common text, code is less linear and often distributed in “patchy” environments (files, modules, etc.) and features such as syntax highlighting in conjunction with specific keywords and symbols provide attention guidance comparable colorful berries. Mismatches between MVT predictions for code and text search can provide cues for how to improve coding environments.

To approach this goal, we derive the following research questions:

RQ1: Does the marginal value theorem predict human search behavior better for code than for ordinary text?

RQ2: Under which conditions does code search behavior resemble text search behavior and under which conditions is it different?

3. Research Proposal

To investigate RQ1 and RQ2, we propose a series of controlled experiments.

Experiment 1: A baseline For the first experiment, we would like to create a baseline foraging scenario where code and text are expected to be processed similarly. We will look at the search for typographical errors in text snippets versus the search for syntax errors in code snippets with and without typical syntax highlighting (see figure 1). For all variants, the snippets are distributed over the screen in order to keep the scenarios both similar and close to a natural foraging task. Clicking the “Next” button moves the participant to the next foraging patch. The participants are instructed to find and click on as many errors as possible in a fixed amount of time. To this end, they are allowed to skip into a new, error-filled patch whenever they feel that searching the current one gets too slow as more and more errors are already discovered.

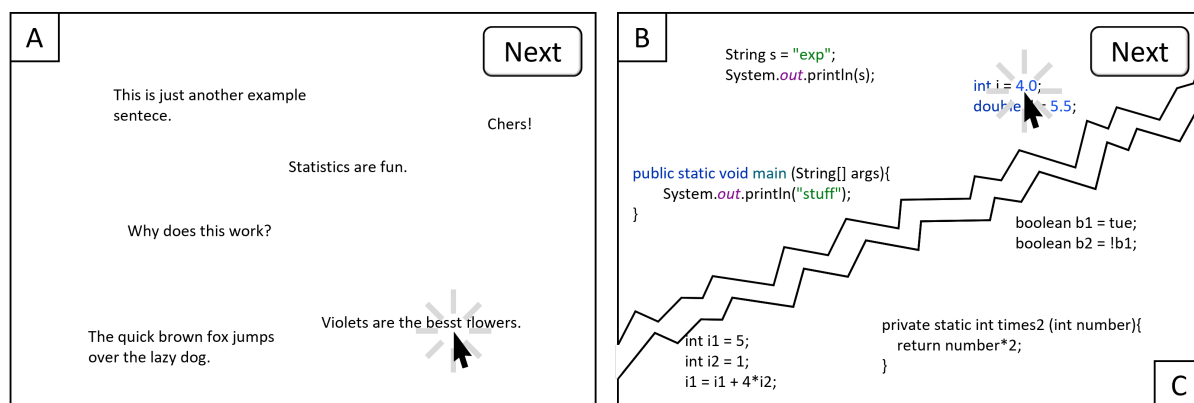


Figure 1 – Exemplary setup for experiment 1.

Over the course of the experiment, the participant is faced with three types of screen setup with an associated search task (“trial”). Trials of type **A** represent the search for typographical errors in text snippets. Trials of type **B** and **C** represent the search for syntactical errors in code snippets with and without syntax highlighting, respectively. Only a few snippets per screen are shown here for readability. Actual patches would be crowded with more elements.

While these tasks are not a perfect mapping, they are similar enough for our purposes. Both tasks are on a relatively superficial level and involve little semantic processing. Any differences between code and text processing detected in this experiment would likely result from the low-level visual differences (e.g.

highlighting). More interestingly, we intend to utilize the results to establish a baseline to be compared to tasks that model more of the distributed nature of program code, the goal of Experiment 2.

Experiment 2: Realistic scenarios Experiment 2 delves into more realistic scenarios: Longer code pieces as well as longer text facilitate more complex search tasks. In this scenario, differences between text and code processing might be more pronounced. Due to the distributed nature of program code, we expect the code search tasks to be better predictable by the marginal value theorem. A stronger difference between code and text in Experiment 2 compared to Experiment 1 could highlight the impact of these structural differences beyond the low-level perceptual differences. In order to eliminate low-level visual differences as the main factor, for text and code both, the deviation from the baseline experiment can be considered.

Variations: These starting experiments offer several connecting points for follow-ups and variations. For instance, different forms of highlighting could be explored. Text highlighting is a popular technique that is used to draw the user's attention to specific text sections or keywords. Text highlighting techniques can also be used for code (Dreyer, Michel, & Bockisch, 2026) which would allow for an interesting very direct comparison. On the other hand, developers are extremely familiar with syntax highlighting as it is not only the default in integrated development environments but also for displaying code snippets in other contexts. Comparing code snippets with syntax highlighting to text snippets with similar structural elements could also be very interesting.

Another factor that is likely to impact performance and which might interact with forms of code presentation is the experience level. Programming novices might behave entirely differently from programming experts. Investigating the search behavior of people without any programming experience could also be interesting in a comparison to people with programming experience.

4. Conclusions and Future Work

We have sketched two experiments that start an investigation of how the optimal foraging framework can be employed to specifically study how programmers search in source code, a specialized form of text search. Extending earlier approaches, we are particularly interested in the interaction of visual perception and informational value. We believe that this approach is insightful and might inform future work on programming tools.

5. References

- Charnov, E. L. (1976, April). Optimal foraging, the marginal value theorem. *Theoretical Population Biology*, 9(2), 129–136. doi: 10.1016/0040-5809(76)90040-x
- Chi, E. H., Gumbrecht, M., & Hong, L. (2007). Visual foraging of highlighted text: An eye-tracking study. In *Human-computer interaction. hci intelligent multimodal interaction environments* (p. 589–598). Springer Berlin Heidelberg. doi: 10.1007/978-3-540-73110-8_64
- Dreyer, T., Michel, J., & Bockisch, C. (2026, March). Towards transferring text comprehension techniques to program comprehension. In *Companion proceedings of the 10th international conference on the art, science, and engineering of programming* (p. 18–25). ACM. doi: 10.1145/3801119.3801126
- Duggan, G. B., & Payne, S. J. (2009). Text skimming: The process and effectiveness of foraging through text under time pressure. *Journal of Experimental Psychology: Applied*, 15(3), 228–242. doi: 10.1037/a0016995
- Grössle, I. M., Schubö, A., & Tünnermann, J. (2023). Testing a relational account of search templates in visual foraging. *Scientific Reports*, 13(1), 12541. doi: 10.1038/s41598-023-38362-9
- Lawrance, J., Bogart, C., Burnett, M., Bellamy, R., Rector, K., & Fleming, S. D. (2013). How programmers debug, revisited: An information foraging theory perspective. *IEEE Transactions on Software Engineering*, 39(2), 197–215. doi: 10.1109/TSE.2010.111

- Niu, N., Mahmoud, A., & Bradshaw, G. (2011, May). Information foraging as a foundation for code navigation (nier track). In *Proceedings of the 33rd international conference on software engineering* (p. 816–819). ACM. doi: 10.1145/1985793.1985911
- Pirolli, P., & Card, S. (1995). Information foraging in information access environments. In *Proceedings of the sigchi conference on human factors in computing systems - chi '95* (p. 51–58). ACM Press. doi: 10.1145/223904.223911
- Pirolli, P., & Card, S. (1999, October). Information foraging. *Psychological Review*, 106(4), 643–675. doi: 10.1037/0033-295X.106.4.643
- Wolfe, J. M. (2013, January). When is it time to move to the next raspberry bush? foraging rules in human visual search. *Journal of Vision*, 13(3), 10–10. doi: 10.1167/13.3.10
- Wolfe, J. M. (2020). Visual search: How do we find what we are looking for? [Journal Article]. *Annual Review of Vision Science*, 6(Volume 6, 2020), 539-562. doi: 10.1146/annurev-vision-091718-015048