Towards Mining Eye-Tracking Datasets for Expertise Prediction

Jenna Wise

Advisors: Dr. Bonita Sharif, CSIS Dept. and Dr. David Pollack, Math & Stats Dept.

Abstract

What We explore the feasibility of using eye gaze data to quantify the expertise of software developers during bug fixing tasks.

How Several sequential analysis techniques were used from TraMineR to analyze developer expertise.

Results Can quantify the expertise of software developers during bug fixing tasks that require changing multiple source code elements for a solution.

Data

Data Collection Goal: To investigate the detailed navigation behavior of developers for realistic change tasks. [1]

Participants: 12 professional developers at ABB Inc. and 10 computing students at Youngstown State University.

Tasks: Find and fix three bugs in the JabRef repository based on real-world bug reports.

Data Collection Tool: iTrace, an Eclipse plugin, works with an eye-tracker to capture eye gaze fixation data on source code elements.

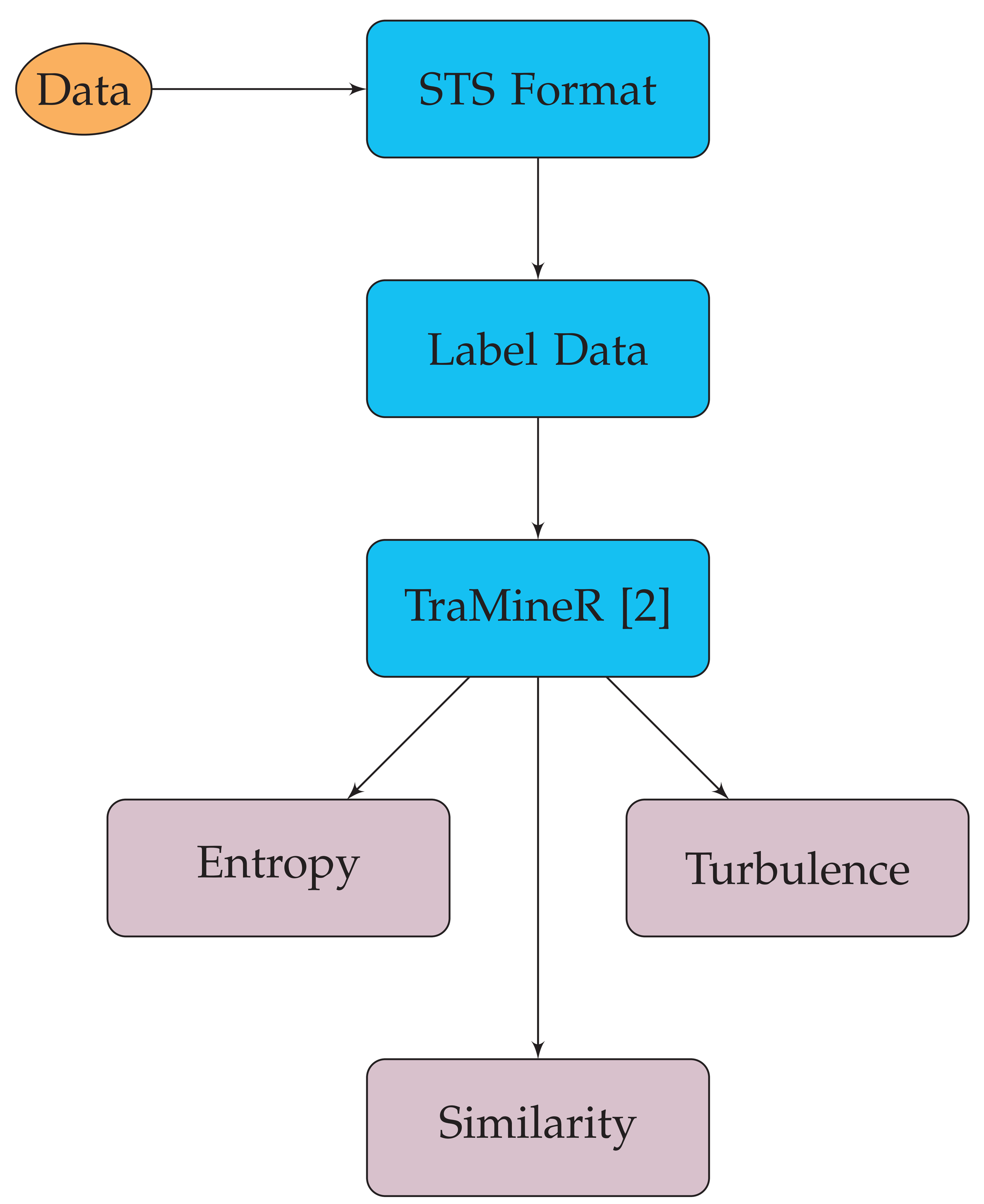
Sequence Format: A fixation contains many data fields, but we only use **fully qualified name** and **duration** to generate sequences in states-sequence format (STS).

A sample of our data in STS format is in Figure 1.

Id	STS Sequence					
24	358	358	358	358	358	359
7	1	1	1	1	1	1

Figure 1: Participants 24 & 7's first 6 source code elements in STS format for Task 1

Experiment



Results

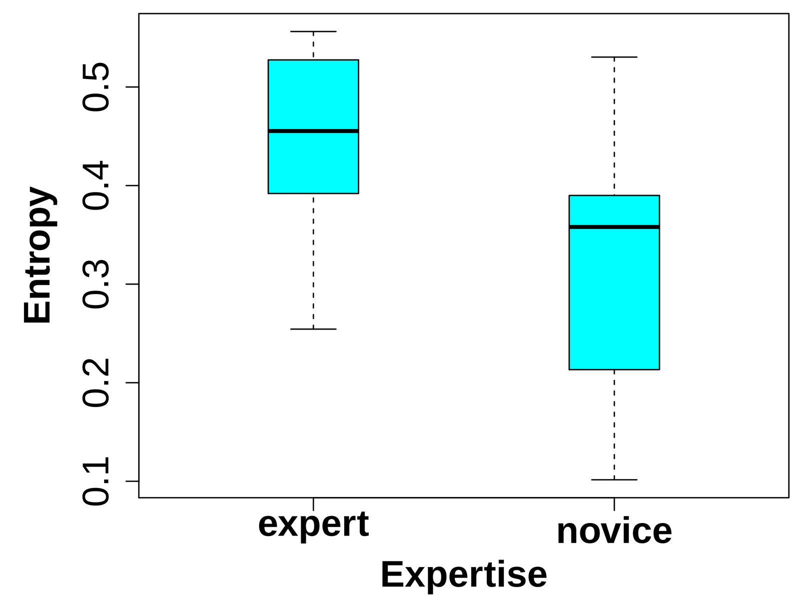


Figure 2: Task 1 entropy

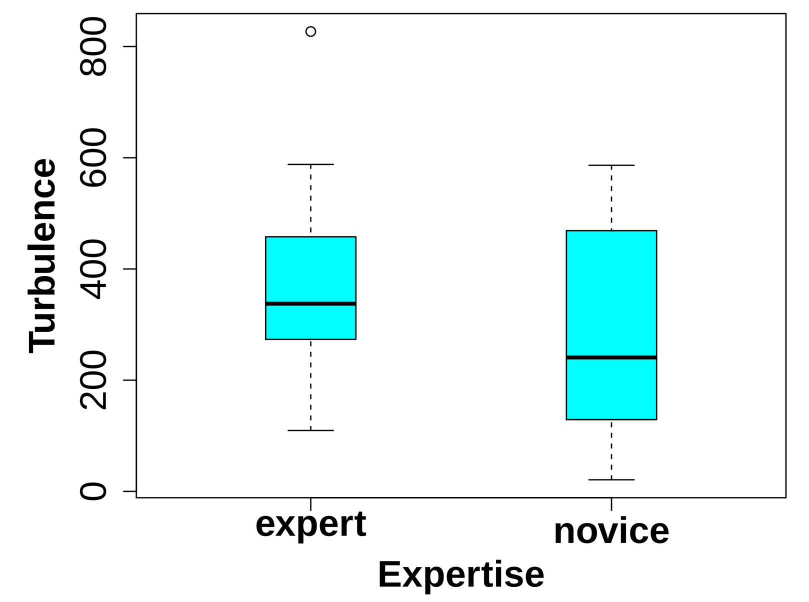


Figure 3: Task 1 turbulence

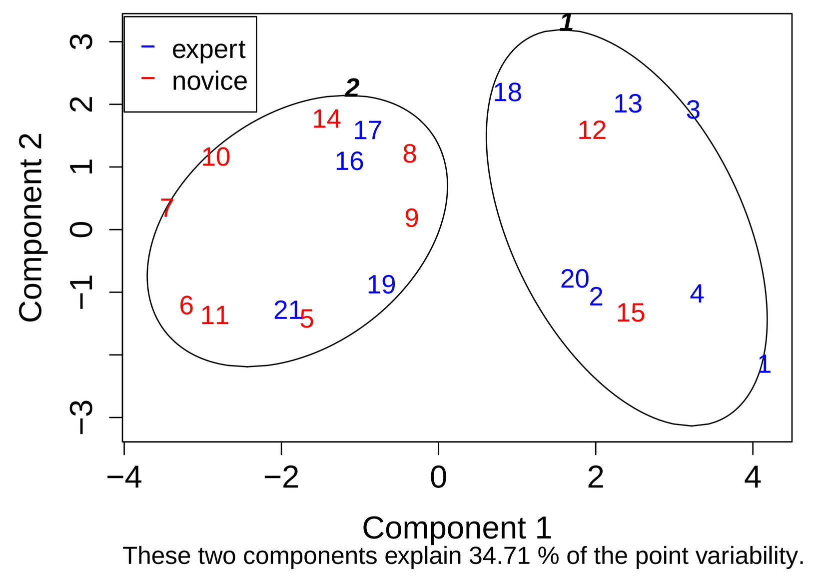


Figure 4: Task 1 LCS metric k-means 2 cluster

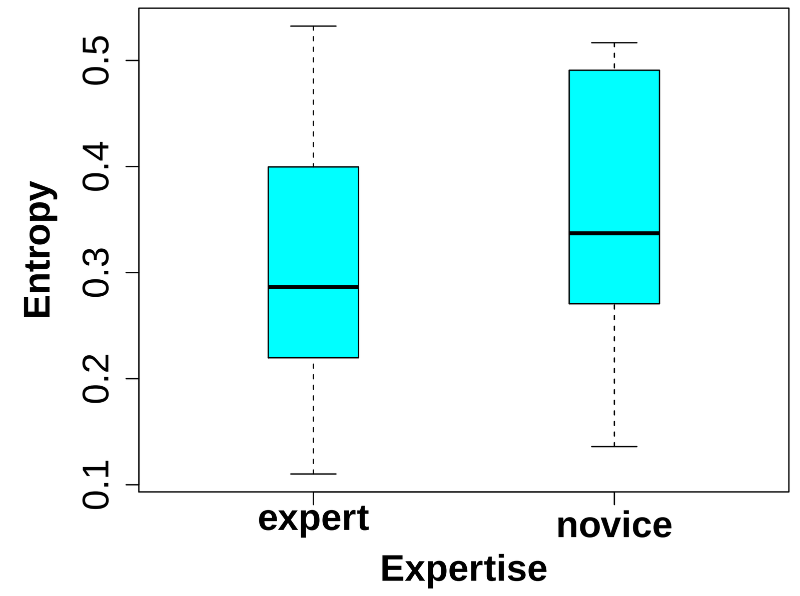


Figure 5: Task 2 entropy

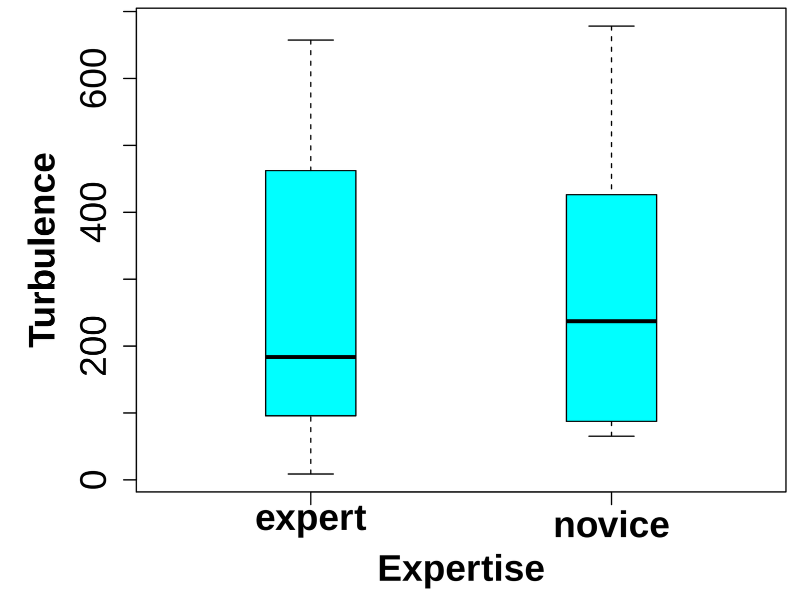


Figure 6: Task 2 turbulence

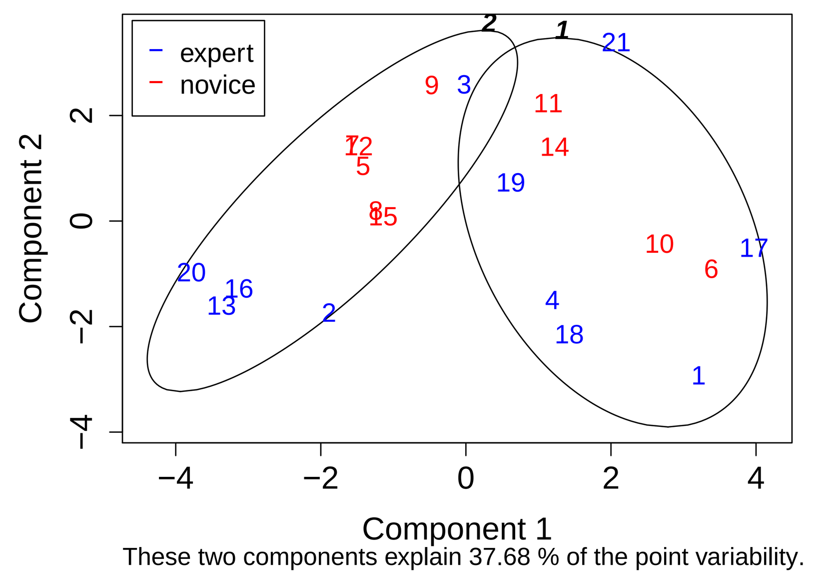


Figure 7: Task 2 LCS metric k-means 2 cluster

Conclusions

- The more a bug requires fixing code across multiple methods, classes, and files (Task 1) the more distinct expert and novice eye gaze sequences are.
- Results were mixed for Tasks 2 and 3 without clear distinctions between experts and novices. Entropy and turbulence patterns found in Task 1 were contradicted in these tasks.
- Out of the three similarity metrics we used, Longest Common Subsequence and Optimal Matching clustered well for Task 1 and Longest Common Prefix clustered poorly over all tasks due to its dependence on the longest common prefix of two sequences.

Future Work

- Analyze sequences of visited source code lines per participant for the most viewed methods in each task.
- Use other sequential analysis techniques to explore the subsequences of source code elements that define the similarity metric clusters.
- Perform further statistical tests on entropy and turbulence values.

Contact Information

Name Jenna Wise
Email jlwise@student.ysu.edu

Name Bonita Sharif
Email bsharif@ysu.edu

Name David Pollack
Email dhpollack@ysu.edu

References

[1] Katja Kevic, Braden M Walters, Timothy R Shaffer, Bonita Sharif, David C Shepherd, and Thomas Fritz. Tracing software developers' eyes and interactions for change tasks. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 202–213. ACM, 2015.

[2] Alexis Gabadinho, Gilbert Ritschard, Matthias Studer, and Nicalas S Müller. Mining sequence data in r with the traminer package: A users guide for version 1.2. *Geneva: University of Geneva*, 2009.