SAND2022-16142C

PSYCHONOMIC SOCIETY
63RD ANNUAL MEETING

A psycholinguistic analysis of eye-movement patterns while troubleshooting Python source code

Jack Dempsey, Anna Tsiola, Nigel Bosch, Kiel Christianson, & Mallory Stites

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory ... Inc. for the

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

A Literacy Perspective for Programming

- Source code primarily human-to-computer interaction
  - But written with human-to-human interaction in mind (Knuth, 1984)

- Dual channel constraints theory for reading code (Casalnuovo et al., 2020)
  - Algorithm channel → knowledge of programming language
  - Natural language channel → knowledge of human language

- Source code syntax less flexible
  - Natural language comprehension uses heuristics
    - Good-Enough Processing, Noisy Channel updates, etc.

Structure of a Python Function

- Function words – carry out specific task
  - ‘len()’ counts number of characters in string

- Content words – arbitrarily established referents
  - ‘word’ represents string being iterated on
  - could also be ‘psychonomics’

```
def is_palindrome(word):
    #This function determines whether a word is the same spelled backwards (i.e. a palindrome).
    #Example: Input: 'racecar'
    #         Output: True
    order = []
    rev_order = []
    for i in range(len(word)):
        order.append(word[i])
        rev_order.append(word[-(i+1)])
    if order == rev_order:
        return True
    else:
        return False
```

Reading Source Code

- Predictability effects (overt judgments)
  - Comprehenders prefer more predictable code (Casalnuovo et al., 2020bc)
  - Code is more predictable than natural language (Casalnuovo et al., 2019, 2020abc)

- Source code comprehension → subset natural text comprehension mechanism, may differ in certain aspects (Fedorenko et al., 2019)
  - Need to compare reading patterns of natural text to reading patterns of source code

Aims of Current Study

1. Provide descriptive analysis of eye-movements in Python source code debugging
    - Test whether global reading patterns predict successful bug detection rate

2. Investigate reading patterns associated with bug detection
    - Test whether these reading patterns predict successful bug detection rate

Experiment Overview (N=30)

- 21 items
  - No Bug Condition
  - Semantic Bug Condition
  - Syntactic Bug Condition
- Commented Out Lines
  - Instructions
  - Input
  - Output

ITEM VARIABLE	BREAKDOWN
NUMBER OF LINES	Mean = 14.8 Lines SD = 3.0 Lines
NUMBER OF INTEREST AREAS	Mean = 23.7 IAs SD = 6.1 IAs
AVERAGE INTEREST AREA COMPLEXITY	Mean = 1.2 Levels SD = 0.1 Levels
AVERAGE INTEREST AREA LENGTH	Mean = 3.2 Characters SD = 0.7 Characters
DISTRIBUTION OF IA TYPES	Object = 42% Logic = 15% Instructions = 14% Loop = 10% Definition & Import = 9% Return = 7% Function = 3%

Experimental Conditions

No Bug

```
def is_palindrome(word):
    #This function determines whether a
    #Example: Input: 'racecar'
    #         Output: True
    order = []
    rev_order = []
    for i in range(len(word)):
        order.append(word[i])
        rev_order.append(word[-(i+1)])
    if order == rev_order:
        return True
    else:
        return False
```

Semantic Bug

```
def is_palindrome\(word):\
#This function determines whether a
#Example: Input: 'racecar'\
#         Output: True\
order =\ []\
rev_order =\ []\
for i in \range(len(word)):\
    order.append(word[i])\
    rev_order.append(word[-(i)])\
if order ==\ rev_order:\
    return True\
else:\
    return False\
```

Syntactic Bug

```
def is_palindrome\(word):\
    #This function determines whether a
    #Example: Input: 'racecar'\
    #         Output: True\
    order =\ []\
    rev_order =\ []\
    for i in \range(len(word)):\
        order.append(word(i))\
        rev_order.append(word[-(i+1)])\
    if order ==\ rev_order:\
        return True\
    else:\
        return False\
```

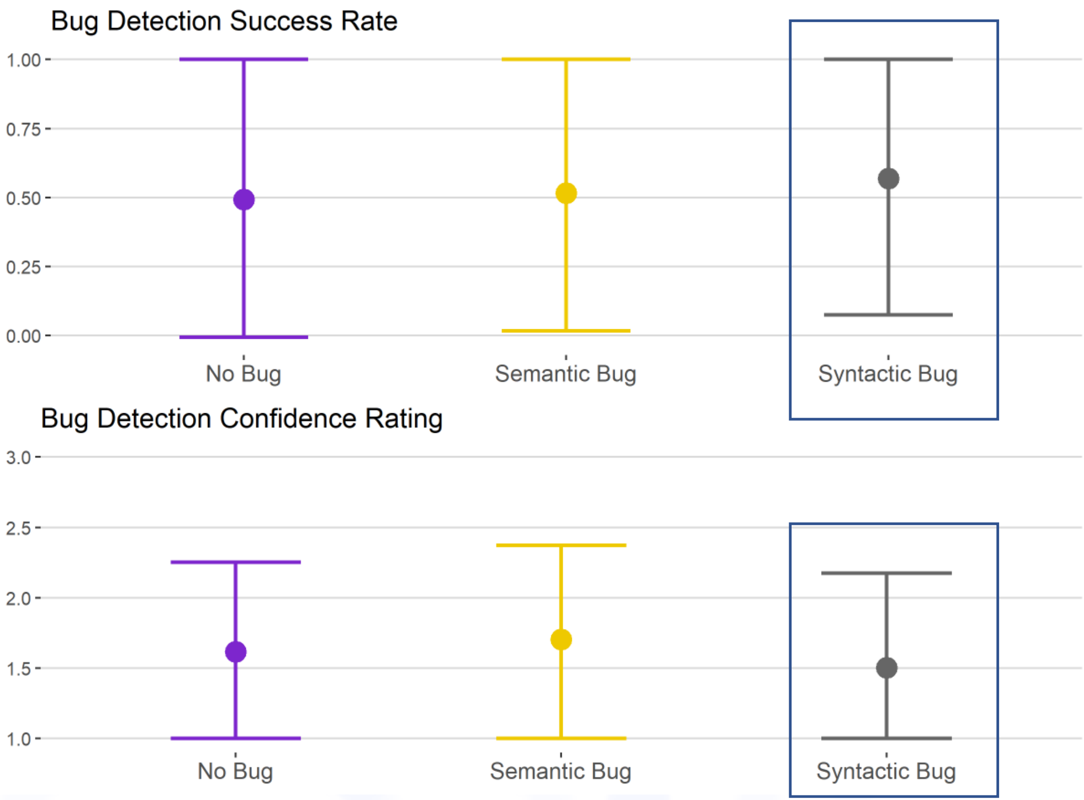
Experiment Procedure

- EyeLink 1000+ Eye-Tracker

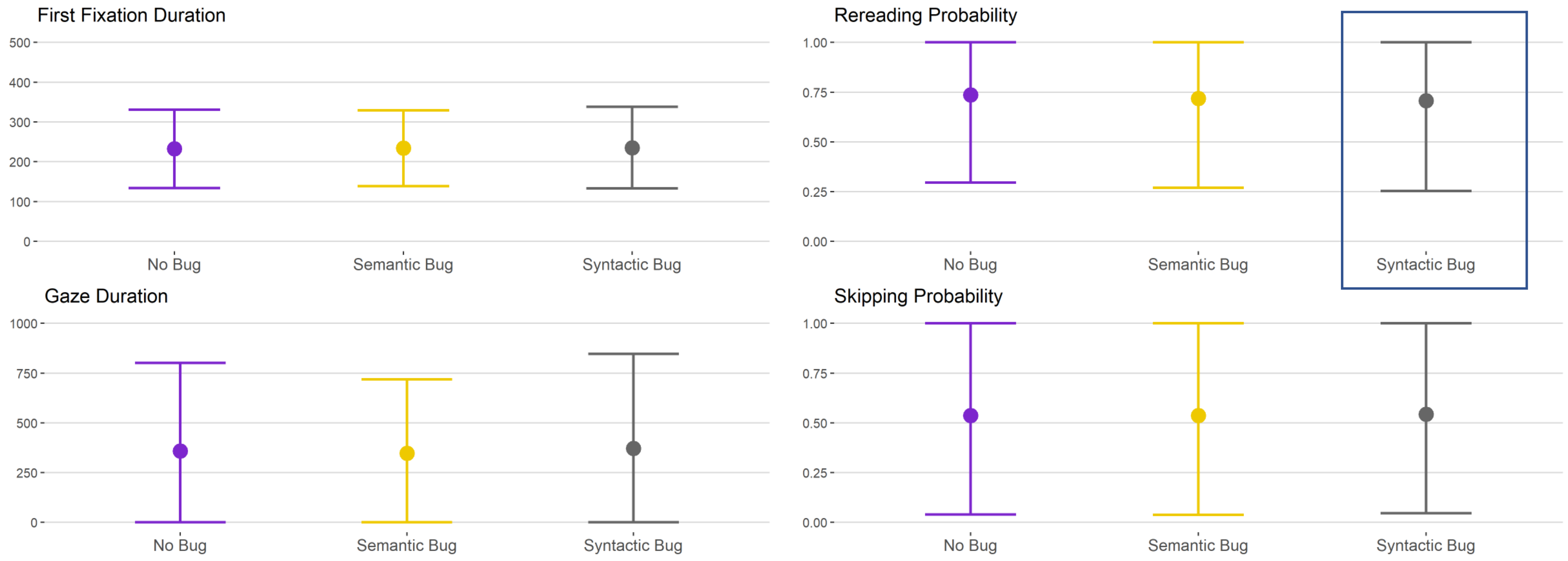
- Once participant made decision, selected
  - No bug
  - Code will run, but give undesired result (semantic bug)
  - Code will yield a runtime error (syntactic bug)

- After decision, participants indicated confidence
  - Extremely confident
  - Somewhat confident
  - Not at all confident

Descriptive Analysis – Accuracy & Confidence



Descriptive Analysis – Global Reading

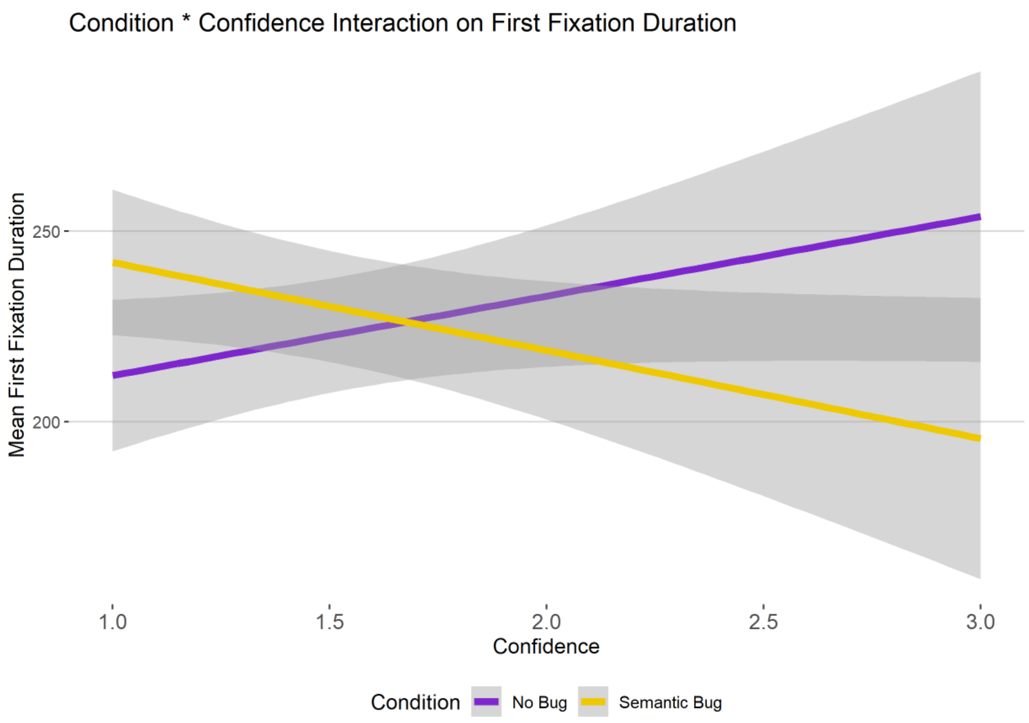


Bug Detection Analysis

- Length = # Characters in IA
- Complexity = # Embeddings in IA

- As IA length & complexity increase across conditions
  - IA Skipping decreases
  - IA GD increases
  - IA rereading increases

- Condition
  - Longer GD in semantic trials
  - Confidence * Condition



#psynom22

Discussion

- Length and complexity effects similar to natural text reading
  - Same general mechanisms likely underlie source code reading
  - Natural language channel (Casalnuovo et al., 2020)

- Confidence associated with initial processing difficulty?
  - Longer early processing associated with higher confidence for no bug trials
  - Faster early processing associated with lower confidence for semantic bug trials

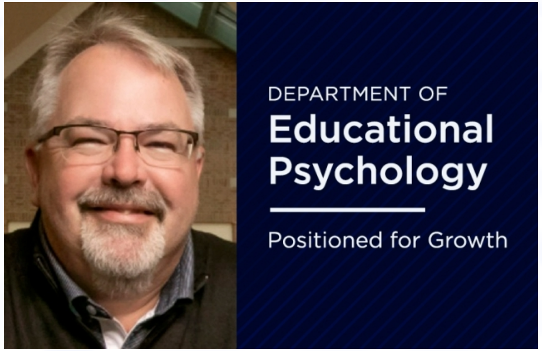
Future Work & Conclusions

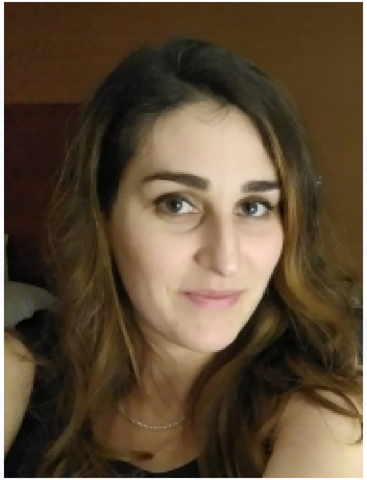
- Reading while debugging Python source code
  - Uses at least some subset of reading for comprehension system
  - Requires knowledge of non-linguistic syntactic system
- What elements of code are particularly troublesome?
  - Moving away from semantically opaque object labels
- More systematic examinations of bug detection behaviors
  - Understand cognitive processes
  - Develop educational interventions & troubleshooting software

Thank you!

Sandia National Laboratories

BECKMAN INSTITUTE





#psynom22

Eye-Movement Patterns During Natural Reading

- Reading for comprehension
  - ~ 20% words skipped
  - First fixation durations ~ 220 msec
  - Rereading probability ~ 5%

- Reading for proofreading
  - Spelling errors → nonwords
    - ~ 10% skipped, FFD ~ 280, ~ 28% rereading probability
  - Spelling errors → wrong words
    - ~ 10% skipped, FFD ~ 265, ~ 45% rereading probability

Schotter, Bicknell, Howard, Levy, & Rayner (2014)

Participants

PARTICIPANT VARIABLE	BREAKDOWN
AGE	Mean = 21.7 SD = 4.2
GENDER	27 Males 3 Females
ETHNICITY	12 South/Southeast Asian 8 Asian/East Asian 7 White 3 Hispanic
MAJOR FIELD OF STUDY	18 Computer Science/Engineering 5 Engineering 3 Statistics/Math 3 Physics/Biophysics 1 Psychology
LANGUAGE	29 Native English 1 ESL
PROGRAMMING EXPERIENCE	Mean = 4.8 Years SD = 2.1 Years
PYTHON EXPERIENCE	Mean = 3.2 Years SD = 1.5 Years
PYTHON COMFORT	Mean = 4.1 SD = 0.7