PROGRAMMATICALLY IDENTIFYING COGNITIVE BIASES PRESENT IN SOFTWARE DEVELOPMENT

Amanda E. Kraft, Matthew Widjaja, Trevor M. Sands, Brad J. Galego

LOCKHEED MARTIN

THE PROBLEM & DATASETS

- Artificial Intelligence- (AI) and Machine Learning- (ML) based systems are increasingly supporting decision-making but are also being questioned for its objectivity.
- Though efforts are underway to make AI/ML systems more explainable and debias datasets, little research is directed at the cognitive biases that developers unintentionally introduce while developing software.
- We manually collated data from two internal codebases and an open-source dataset from refactoring tasks on various open-source projects.

Internal Project A

1405 Commit Messages

Internal Project B

227 Commit Messages
469 Code Comments
181 Code Docstrings

Open-Source Code Smell [6]

441 Commit Messages
https://smilevo.github.io/self-affirmed-refactoring/IWoR19_index.html

COGNITIVE BIAS TYPES

Cognitive Biases are systematic deviations from cognitive processes, and hundreds have been identified [1].
40 of those biases have been investigated in Software Engineering [2] and we selected four of those to focus on.

Anchoring Bias

Tendency to rely too heavily on pre-existing or first information found when making a quantitative judgement. [2]

Confirmation Bias

Tendency to search for & focus on information that confirms one preconception(s) while ignoring or rejecting sources that challenge it. [4]

Availability Bias

Tendency to overestimate the likelihood of events based on the ease of which examples come to mind. [3]

Hyperbolic Discounting

Tendency to prefer immediate payoffs over larger rewards at a later point. [2]

Subjective/Other

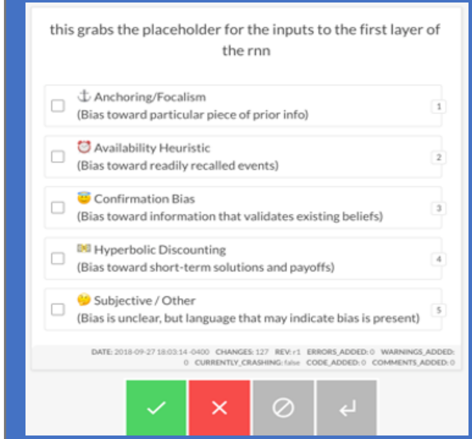
This is not an official type of cognitive bias but was used on this project to categorize biases that weren't listed above.

DATA CURATION & ANNOTATION PROCESS

Raw Data

- Project A Commits
- Project B Commits,
- Project B Comments
- Project B Docstrings
- Code Smell Commits

A group of 5-6 reviewers independently reviewed each entry for cognitive bias (if any) using Prodigy



this grabs the placeholder for the inputs to the first layer of the rnn

- □ ⚓ Anchoring/Focalism (Bias toward particular piece of prior info) 1
- □ ⏰ Availability Heuristic (Bias toward readily recalled events) 2
- □ 😇 Confirmation Bias (Bias toward information that validates existing beliefs) 3
- □ 💵 Hyperbolic Discounting (Bias toward short-term solutions and payoffs) 4
- □ 🤔 Subjective / Other (Bias is unclear, but language that may indicate bias is present) 5

DATE: 2018-09-27 18:03:14 -0400 CHANGES: 127 REV: r1 ERRORS_ADDED: 0 WARNINGS_ADDED: 0 CURRENTLY_CRASHING: false CODE_ADDED: 0 COMMENTS_ADDED: 0

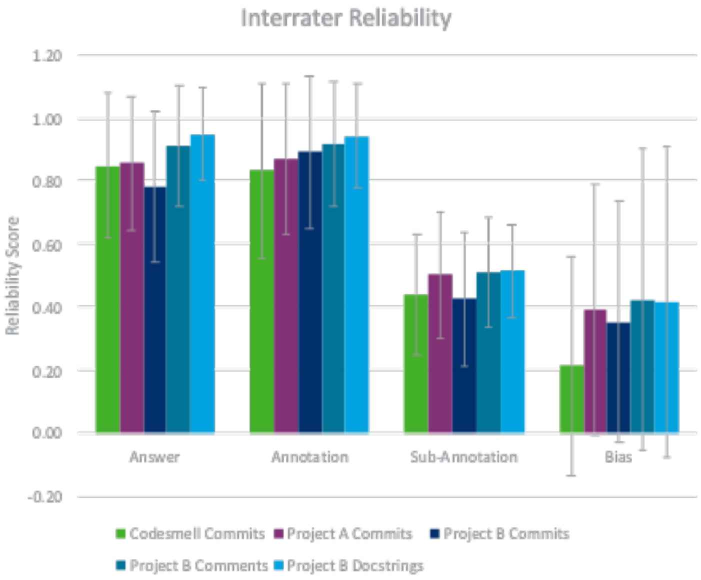
The green ‘accept’ button meant bias was found. The red ‘reject’ button meant that no bias was found. The grey ‘ignore’ meant that the message had no meaningful content.

A reviewer finalized the labels, selecting one bias type per entry.

Interrater reliability (IRR) was rather low in terms of specific biases found. While reviewers typically agree if an entry is or isn’t biased, reviewers tend to disagree on which bias is present. More details about the IRR process is in our paper.

This table visualizes the interrater reliability across datasets. Error bars show standard deviation in the reliability scores.

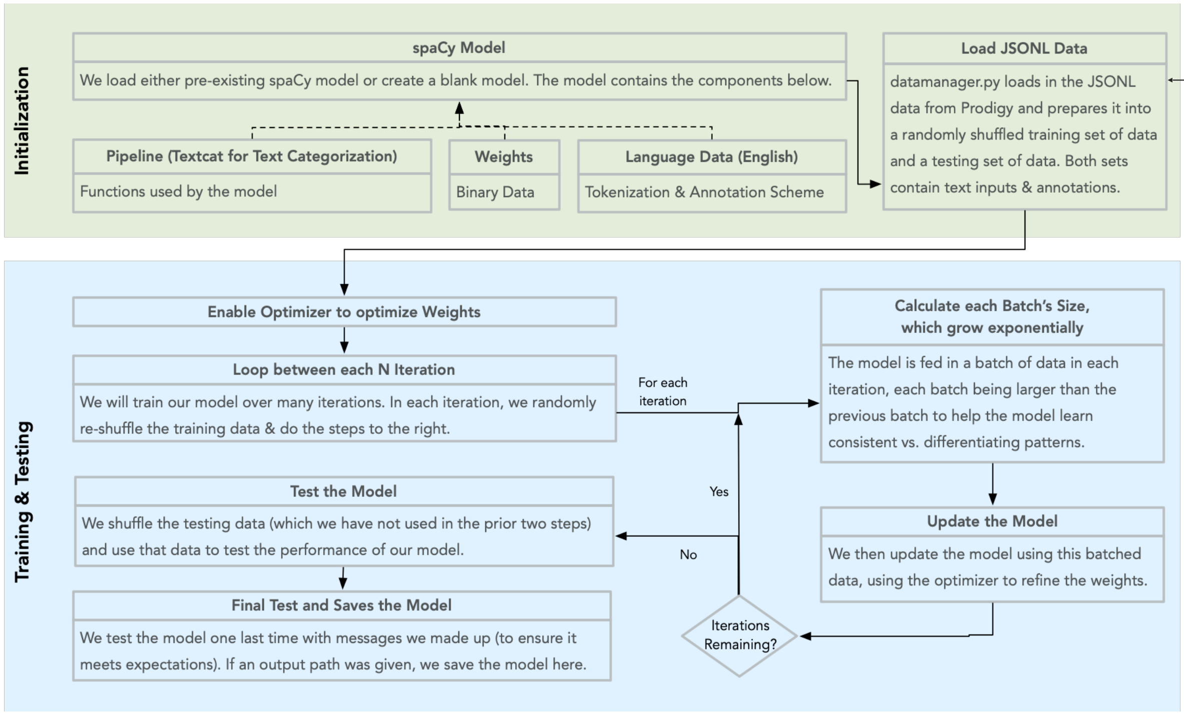
Reliability was higher for the initial answer on if an entry was or wasn’t biased, but they dropped when reviewers had to select the specific bias type. More information is on page 3 of the paper.



LOCKHEED MARTIN

MODELING PROCESS

The final dataset was ingested into a spaCy text classification model.



spaCy Model

We load either pre-existing spaCy model or create a blank model. The model contains the components below.

Pipeline (Textcat for Text Categorization)

Functions used by the model

Weights

Binary Data

Language Data (English)

Tokenization & Annotation Scheme

Load JSONL Data

datamanager.py loads in the JSONL data from Prodigy and prepares it into a randomly shuffled training set of data and a testing set of data. Both sets contain text inputs & annotations.

Enable Optimizer to optimize Weights

Loop between each N Iteration

We will train our model over many iterations. In each iteration, we randomly re-shuffle the training data & do the steps to the right.

Calculate each Batch's Size, which grow exponentially

The model is fed in a batch of data in each iteration, each batch being larger than the previous batch to help the model learn consistent vs. differentiating patterns.

Update the Model

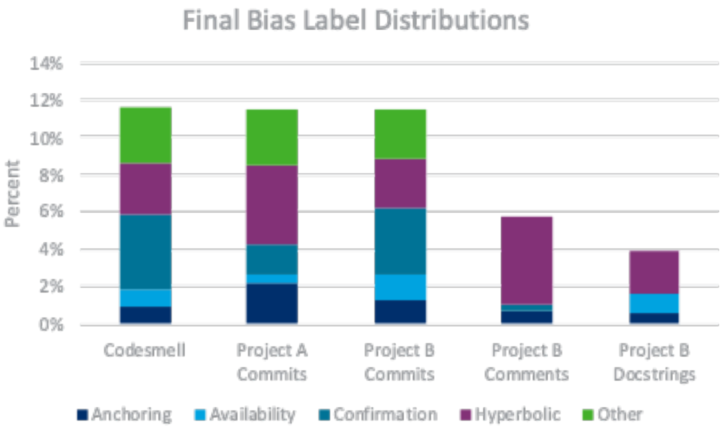
We then update the model using this batched data, using the optimizer to refine the weights.

Test the Model

We shuffle the testing data (which we have not used in the prior two steps) and use that data to test the performance of our model.

Final Test and Saves the Model

We test the model one last time with messages we made up (to ensure it meets expectations). If an output path was given, we save the model here.



This plot shows the distribution of the final cognitive bias labels from each dataset our team annotated. This data is ingested into our models, either per codebase or as a combined dataset.

MODELS RAN + MODEL RESULTS

The spaCy text classification model was configured so it could either predict for the presence of bias (binary) or for the specific type of cognitive bias found.

Binary Models

Models which identified whether text entries contained bias performed better, averaging 79-83% F1.

This aligns with how human raters tended to agree whether an entry is biased or not.

Project B Commits' performed poorer, which correlates with the smaller size of the dataset.

Dataset	Total Items	Mean F1	Std. Dev
Project A Commits	1405	81.2%	2.6%
Project B Commits	227	65.9%	14.0%
Project A + B Commits	1632	79.0%	5.1%
Project B Commits + Comments	696	78.6%	6.8%
All Data	2282	82.3%	3.9%

Multi-Label Models

Models which identified the specific type of cognitive bias performed poorer, in line with how human raters tended to disagree on how a specific entry is biased.

Dataset	Total Items	Mean F1	Std. Dev
Project A + B Commits	1632	72.1%	5.8%

REFERENCES

[1] Delgado-Rodriguez, M., & Llorca, J. (2004). Bias. Journal of Epidemiology & Community Health, 58(8), 635-641. 10.1136/jech.2003.008466

[2] Mohanani, R., Salman, I., Turhan, B., Rodríguez, P., & Ralph, P. (2018). Cognitive biases in software engineering: a systematic mapping study. IEEE Transactions on Software Engineering, 46(12), 1318-1339. 10.1109/TSE.2018.2877759

[3] Stacy, W., & MacMillan, J. (1995). Cognitive bias in software engineering. Communications of the ACM, 38(6), 57-63. 10.1145/203241.203256

[4] Calikli, G., & Bener, A. (2015). Empirical analysis of factors affecting confirmation bias levels of software engineers. Software Quality Journal, 23(4), 695-722. 10.1109/ICIST.2013.6747696

[6] AlOmar, E., Mkaouer, M. W., & Ouni, A. (2019, May). Can refactoring be self-affirmed? an exploratory study on how developers document their refactoring activities in commit messages. In 2019 IEEE/ACM 3rd International Workshop on Refactoring (IWoR) (pp. 51-58). IEEE. 10.1109/IWoR.2019.00017

WHO WE ARE

This project was developed by the Intelligent Systems Works (ISW) team, within Lockheed Martin's Advanced Technology Laboratories. ISW focuses on research and development to create technologies that will accelerate the hybrid existence of human and machine intelligence.

The ISW team is multidisciplinary and includes over 75 scientists and engineers with advanced degrees in computer science, electrical & mechanical engineering, system engineering, mathematics, cognitive science, neuroscience/psychology, materials science and biology.

The ISW team, along with its academic and industry partners, is leading multiple contracts in the US Government and technology organizations, infusing and augmenting human capabilities with machine intelligence.

To learn more or ask questions, email brad.j.galego@lmco.com and/or matthew.widjaja@lmco.com

LOCKHEED MARTIN