AI/ML IRL

Joshua Eckroth
Chief Architect / Assistant Professor of Computer Science
i2k Connect / Stetson University

Certain, closed systems:

- Well-defined inputs (e.g., bounded integers)
- Well-defined transformations and calculations

Certain, open systems:

- Open-ended inputs (e.g., user comments, speech, selfies, etc.)
- Well-defined transformations and calculations

Uncertain systems:

- **Open-ended inputs**
- **Uncertain transformations and calculations**

Do you “need” AI because you want to have AI?

Do I need AI?

- **Online learning / planning**
- **Processing or generating forms of human communication**
- **Detecting or recognizing natural or human artifacts**
- **Encoding expert knowledge**
- **Fast, heuristic, "satisficing" search in massive search spaces**
- **Physical presence, operating among humans or the natural world**

AI = TECHNICAL DEBT

Machine Learning: The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young
`{dsculley,gholt,dgg,edavydov}@google.com`
`{toddphillips,ebner,vchaudhary,mwyoung}@google.com`
Google, Inc

Abstract

Machine learning offers a fantastically powerful toolkit for building complex systems quickly. This paper argues that it is dangerous to think of these quick wins as coming for free. Using the framework of *technical debt*, we note that it is remarkably easy to incur massive ongoing maintenance costs at the system level when applying machine learning. The goal of this paper is highlight several machine learning specific risk factors and design patterns to be avoided or refactored where possible. These include boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, changes in the external world, and a variety of system-level anti-patterns.

“[M]achine learning packages have all the basic code complexity issues as normal code, but also have a larger system-level complexity that can create hidden debt.

Thus, refactoring these libraries, adding better unit tests, and associated activity is time well spent but does not necessarily address debt at a systems level.”

- Machine learning may “subtly erode abstract boundaries”
  - i.e., using outputs from a predictor in other parts of code
  - This creates a tight coupling
- ML is often treated as a black box
  - Results in lots of glue code and calibration code that locks in assumptions
- Changes in external world may make behavior change in unintended ways
  - Ratcheting up maintenance costs
- Monitoring that the system is performing as intended may be difficult
  - Need careful design to put bounds on the chaos a bad ML model can cause

ADVICE

1. **Develop a set of "ground truth" examples**
    - These can be unit tests!
    - Can also serve as a performance metric (% accuracy, etc.)
2. **Add safeguards**
    - Check AI's output for dangerous conditions (bad words, extreme values, etc.)
3. **Limit the impact of AI on the system**
    - AI output can be unpredictable; don't depend on a small range of outputs
    - Other code may break if it makes wrong assumptions about AI output
4. **Provide an alternative UI pathway**
    - Support disabling Clippy
    - Show data that was deemed "irrelevant" in some less-visible menu
5. **Be very cautious about making assumptions for people**
    - You could be wrong
    - People *hate* when your system ignores their intentions
6. **Be very *very* cautious about online learning**
    - At least pay attention to the system – it could be evolving in the wrong direction
7. **Don't put your learning bot on Twitter**
    - i.e., don't let the larger world influence your AI system in a feedback loop
    - Just don't.

AI API'S

Kitchen sink APIs

- **Microsoft Project Oxford**
  - vision: image categories, thumbnail generation (with proper crop), OCR, face detection and recognition, emotion recognition; video: face detection and tracking, auto stabilization, motion detection; speech: voice commands, speaker recognition; language: parse natural language queries
- **IBM Watson**
  - text: keyword extraction, sentiment analysis, named entity recognition, translation; speech: tts, recognition; vision: find objects, people, text in images

Vision APIs

- **Google Vision**
  - concept extraction, face/object detection and recognition, detect offensive content, image sentiment analysis
- **Clarifai**
  - concept extraction (tagging) from images and video

Voice / natural language APIs

- **Api.ai**
  - voice interface; text-to-speech (tts), natural language queries (text or sound), supports user-defined synonyms
- **Wit.ai**
  - voice interface; very similar to Api.ai
- **MindMeld**
  - voice interface; integrates with existing speech recognition (e.g., in ios); supports defining custom “knowledge graph” that matches commands specific to your app
- **Chatbots.io**
  - chatbots, customer care agent bots

AI LIBRARIES

Javascript AI libraries

- **Good luck**
- There are some weak candidates:
  - www.npmjs.com/search?q=artificial-intelligence
- Most AI work requires a model training phase
  - This is computationally expensive
  - A mobile device or website won't do model training
  - Why do this in Javascript?
- Better: create a backend web-service to host the AI
  - Use a language with a solid AI library: Python, Java, C++, etc.

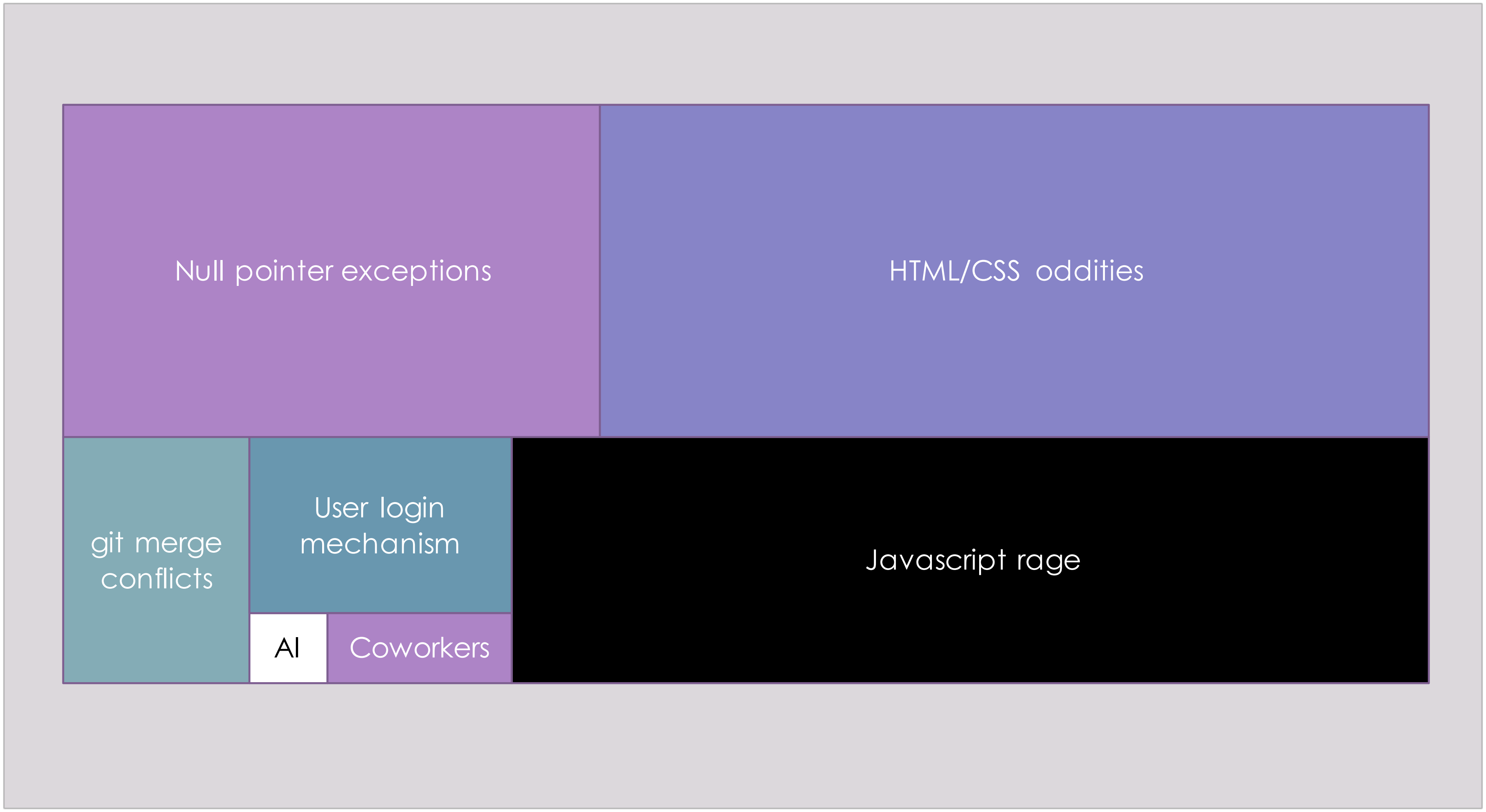
Python AI libraries

- **scikit-learn** (scikit-learn.org)
  - Classification: identify the category of text, images, …
  - Regression: predict a value, such as a stock price, weather, user ratings, …
  - Clustering: find natural groupings to predict user preferences, document similarity, …
- **Theano** (deeplearning.net/software/theano)
  - Deep learning: tag photos, identify faces, extract meaning from text, voice search, speech transcription, …

Java AI libraries

- **Weka** (www.cs.waikato.ac.nz/ml/weka)
  - Classification
  - Regression
  - Clustering
  - Spiffy GUI for trying out various techniques before writing code
- **OpenNLP** (opennlp.apache.org)
  - Text processing: finding names, organizations, dates, etc.
- **Deep Learning for Java** (deeplearning4j.org)

SUMMARY


Null pointer exceptions
HTML/CSS oddities
git merge conflicts
User login mechanism
AI
Coworkers
Javascript rage