Software Engineering

(9) State-of-the-Art Topics

Sokendai / National Institute of Informatics
Fuyuki Ishikawa / 石川　冬樹
f-ishikawa@nii.ac.jp / @fyufyu
http://research.nii.ac.jp/~f-ishikawa/

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

TOC

AI for SE?

- New directions appeared with the AI trend with deep learning in late 2010's
  - e.g., bug prediction of program code
- Advanced automation techniques had been investigated a lot before it
  - e.g., use of metaheuristics optimization or evolutionary computation such as genetic algorithms has been popular for these 10 years (called "search-based software engineering")
  - Showing some of them (maybe biased)

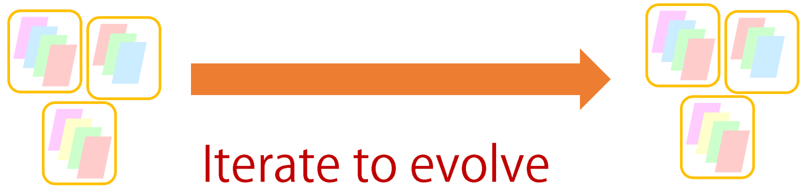
Active Research Area (1) Search-Based SE

- Search-Based Software Engineering:
  - Solve problems in software engineering by reducing them to optimization problems, i.e., define some “score” and maximize it
  - Often done with metaheuristic algorithms
  - Test generation, test minimization, test prioritization, design, requirements selection, bug fix creation, …

Example: Search-Based Testing

- Example: automated test input generation
  - For the unit testing of the method `m1(int x, int y)`, prepare the input vector `(x0, y0), (x1, y2), …, (xn, yn)`
  - Maximize the branch coverage, while minimizing the number of test cases `n`

Solutions starting with random test cases



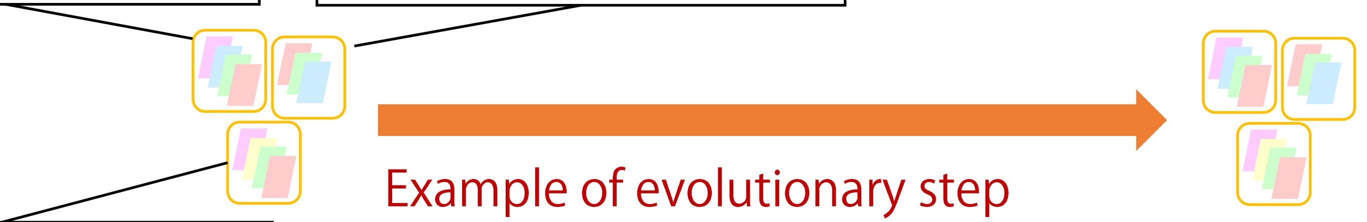
[https://www.evosuite.org/]
[Fraser et al., Whole Test Suite Generation, TSE'13]

Application Example: Mobile App Testing at Facebook

- Sapienz: automated test generation

Test Suite A
#Test Cases: 25
#Commands: 150
#Detected Crashes: 5

Test Suite B
#Test Cases: 80
#Commands: 500
#Detected Crashes: 63



Test Suite C
#Test Cases: 90
#Commands: 520
#Detected Crashes: 20

Example of evolutionary step

Selection: C is dominated A and B, should be dropped

Crossover: mix test cases from A and those from B

Finally,
"compact test report"
is obtained

[Mao et al., Sapienz：multi-objective automated testing for Android applications, ISSTA'16]
[Alshahwan., et al., Deploying Search Based Software Engineering with Sapienz at Facebook, SSBSE'18]

Active Research Area (2) Fault Localization

- Fault Localization: estimating the bug location(s)
  - Example: Spectrum-based Fault Localization

    → Compare lines executed by passed tests and failed tests

Test record

l.1	l.2	l.3	…	Result
✓	✓		…	PASS
✓		✓	…	FAIL
		✓	…	FAIL
…	…	…	…	…

“Suspiciousness” ranking

Line	Score
l.3	0.8
l.5	0.72
l.9	0.6
…	…

$$\text{Ochiai Metric (line)} = \frac{\#\ Failed\ Test\ Cases\ that\ execute\ the\ line}{\sqrt{\#\ Test\ Cases\ that\ execute\ the\ line \times \#\ Failed\ Test\ Cases}}$$

[Wong et al., A Survey on Software Fault Localization, TSE’16]

Active Research Area (3) Automated Program Repair

- Automated Program Repair
  - Given a test suite and a program, modify the program so that it passes all of the test cases in the test suite
  - Common approach: “generate and validate”
    - e.g., apply (inverse) mutation operators that denote typical bugs, such as changing between < and <=
    - e.g., search for the best operators by evolutionary computation
  - “Found 55 out of 105 bugs for $8 each” (GenProg, 2012)
  - Known issue: test cases are often too weak and accept naïve fixes

[Gazzola et al., Automatic Software Repair: A Survey, TSE’19]
[Goues et al., A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each, ICSE’12]

Active Research Area (4) Repository Mining

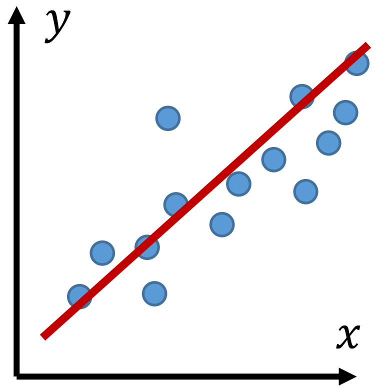
- Mining Software Repositories: MSR （リポジトリマイニング）
  - Make use of data mining techniques
  - e.g., mining typical bug fix patterns, typical rejected pull-requests, etc. from GitHub
  - e.g., extracting wrong API usages or characteristics of sample code from StackOverflow
  - e.g., classifying and mining the comments in app stores

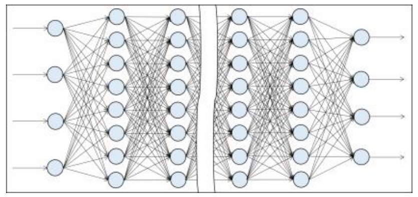
[Kagdi et al., A survey and taxonomy of approaches for mining software repositories in the context of software evolution, JSMR'07]
[http://www.msrconf.org/]

TOC

Machine Learning (Very Briefly)

- Want to predict salary y of a person from his/her age x
  - Assuming the prediction model $y = ax + b$, we can make the function by deciding a, b to fit with the past data
  - Practically, the linear function is not expressive enough
  - In deep learning, we use millions of parameters

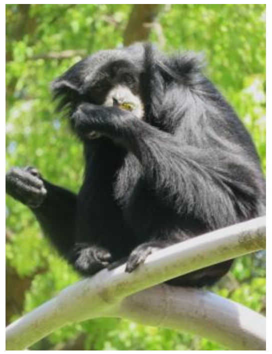




(3rd) Trend of AI mostly with machine learning techniques

Machine Learning: Another Example

Gibbons

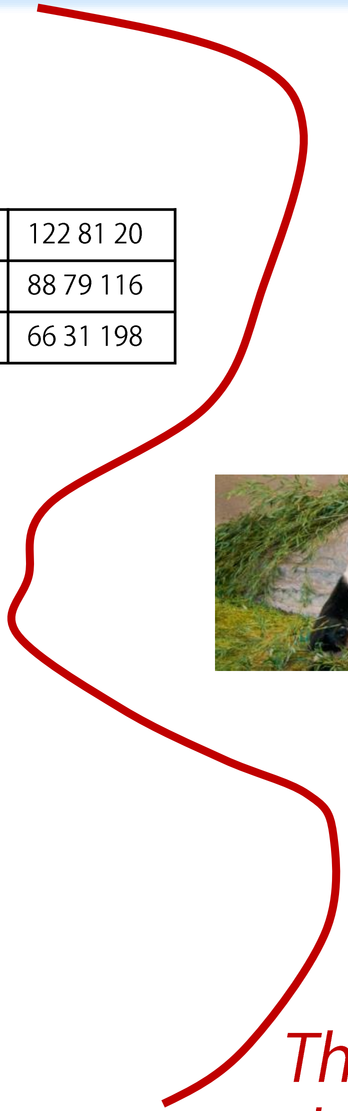


35 24 210	20 121 24	122 81 20
211 54 42	12 222 90	88 79 116
24 36 98	98 181 31	66 31 198

13 83 33	13 45 94	75 74 111
111 8 73	192 1 221	237 31 1
74 35 122	93 76 244	73 211 45

0 245 210	20 12 114	84 99 100
11 86 99	121 88 91	18 0 77
46 87 121	70 76 122	122 14 94

Image =
RGB values for
each pixel



Pandas

254 32 67	222 88 1	108 76 14
12 86 222	98 75 122	111 74 74
198 87 33	188 173 4	68 176 83

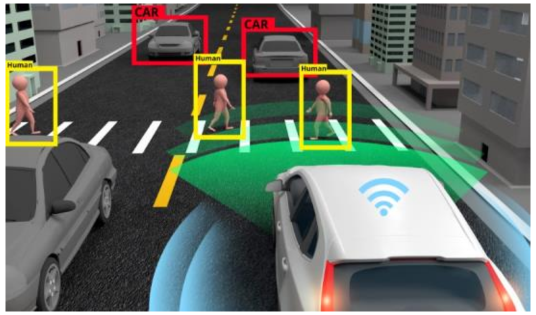
77 81 123	122 158 6	76 63 42
3 3 78	19 183 84	76 63 123
98 83 111	123 7 99	253 48 91

0 24 31	20 21 124	12 101 50
21 54 242	112 22 90	8 79 214
124 56 85	98 99 141	166 1 198

The boundary function is made by the training data

Machine Leaning: What are Great

- Realizing specifications that cannot be rigorously described
  - Object classification in images
    - Pedestrian, signals in driving
    - Anomaly in plants
    - Symptoms in medical images
  - Control signals in plants
  - Loan/insurance approval
  - Various tasks with sentences and voices
  - Generation of images and videos



Machine Leaning: What are Difficult

- Incompleteness
  - No 100% accuracy
  - Accuracy unknown before implementation
- Uncertain behavior
  - No guarantee on the behavior for a new input
  - Often unexplainable
- Dependency on data
  - Requiring large and “proper” training data
  - Evaluation heavily depending on data

Example of Unique Problems: Performance Limitation

- Google photo
  - “Gorilla” for black people
  - No essential fix, inhibiting the term “gorilla”

[https://www.theguardian.com/technology/2015/jul/01/google-sorry-racist-auto-tag-photo-app]

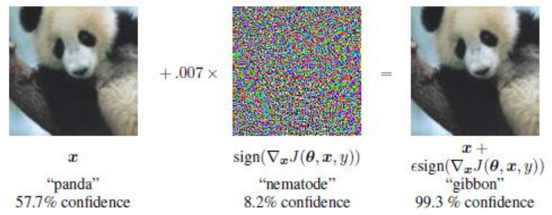
[https://www.theguardian.com/technology/2018/jan/12/google-racism-ban-gorilla-black-people]

- Misrecognition by very small noises

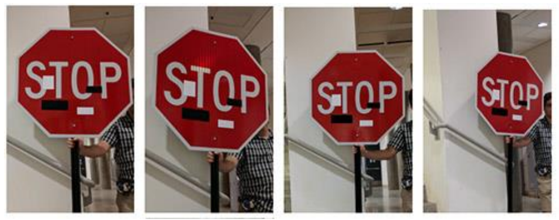
  （adversarial examples・敵対的サンプル）

[Goodfellow et al., Explaining and Harnessing Adversarial Examples, 2015]

[Ackerman, Slight Street Sign Modifications Can Completely Fool Machine Learning Algorithms, IEEE Spectrum’17]


$+ .007 \times$
$=$
x
“panda”
57.7% confidence
$\mathrm{sign}(\nabla_x J(\theta, x, y))$
“nematode”
8.2% confidence
$x + \epsilon\,\mathrm{sign}(\nabla_x J(\theta, x, y))$
“gibbon”
99.3 % confidence

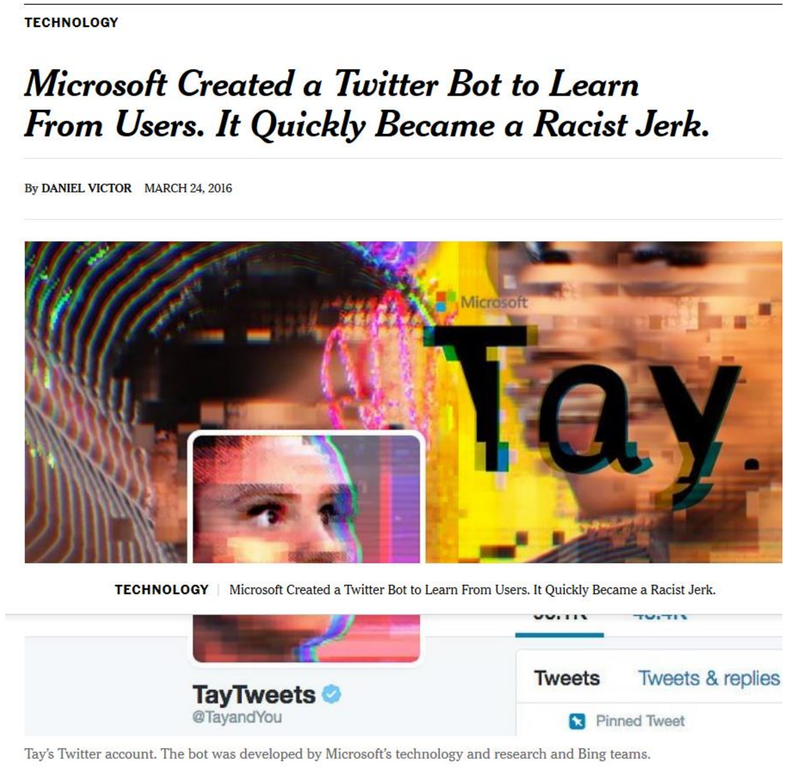
“Panda” to “Gibbon”



Misrecognition by physical tapes

Example of Unique Problems: Continuous Update

- Improper tweets by a Twitter Bot
  - Malicious users guided with discriminatory words
  - Monitoring of continuously updating system (and the update is guided by users)
  - Requirements defined by the society


TECHNOLOGY

Microsoft Created a Twitter Bot to Learn From Users. It Quickly Became a Racist Jerk.

By DANIEL VICTOR MARCH 24, 2016

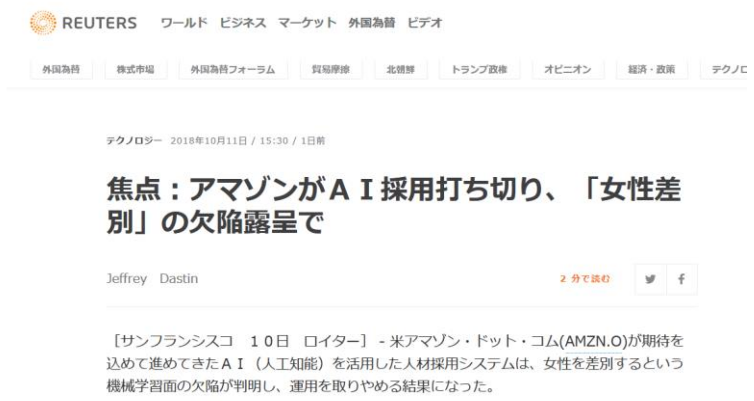
TECHNOLOGY | Microsoft Created a Twitter Bot to Learn From Users. It Quickly Became a Racist Jerk.

Tay's Twitter account. The bot was developed by Microsoft's technology and research and Bing teams.

[https://www.nytimes.com/2016/03/25/technology/microsoft-created-a-twitter-bot-to-learn-from-users-it-quickly-became-a-racist-jerk.html]
(access: 2021/09/27)

Example of Unique Problems: Bias and Ethics

- Occurrence of discriminative biases
  - Learning (unconscious) discriminations
  - Poor prediction for cases with small data


REUTERS ワールド ビジネス マーケット 外国為替 ビデオ

外国為替 株式市場 外国為替フォーラム 貿易摩擦 北朝鮮 トランプ政権 オピニオン 経済・政策 テクノロ

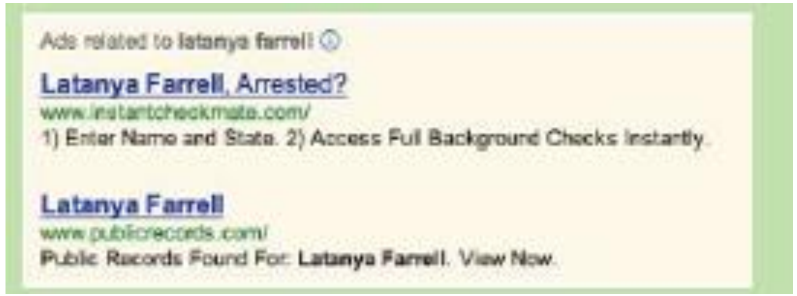
テクノロジー 2018年10月11日 / 15:30 / 1日前

焦点：アマゾンがＡＩ採用打ち切り、「女性差別」の欠陥露呈で

Jeffrey Dastin 2 分で読む

［サンフランシスコ　１０日　ロイター］ - 米アマゾン・ドット・コム(AMZN.O)が期待を込めて進めてきたＡＩ（人工知能）を活用した人材採用システムは、女性を差別するという機械学習面の欠陥が判明し、運用を取りやめる結果になった。

Employment AI by Amazon


Ads related to latanya farrell

Latanya Farrell, Arrested?
www.instantcheckmate.com/
1) Enter Name and State. 2) Access Full Background Checks Instantly.

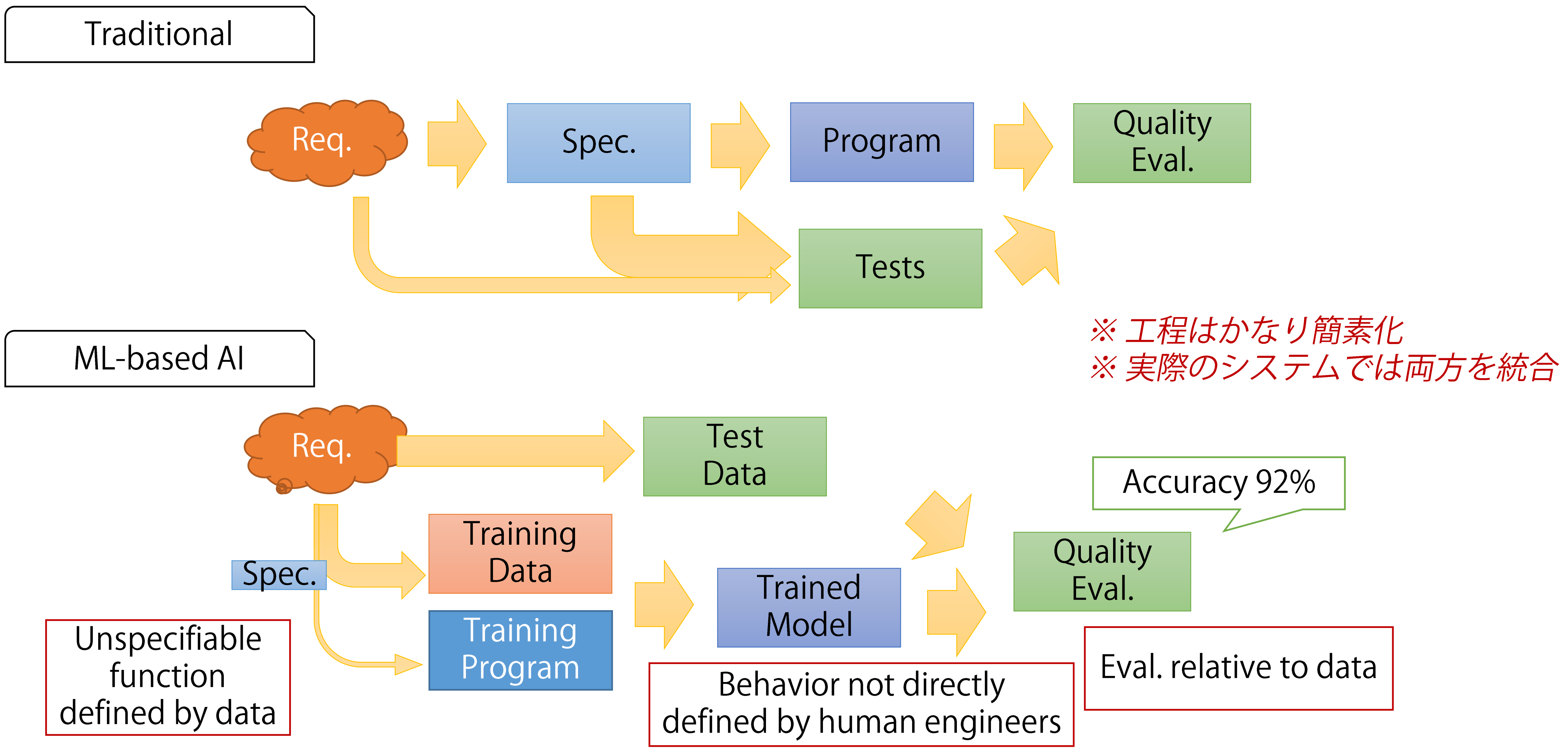
Latanya Farrell
www.publicrecords.com/
Public Records Found For: Latanya Farrell. View Now.

Ad recommendation by Google search biased with search with African names

[https://jp.reuters.com/article/amazon-jobs-ai-analysis-idJPKCN1ML0DN] (access: 2021/09/27)

[L. Sweeney, Discrimination in Online Ad Delivery, ACM Queue'13]

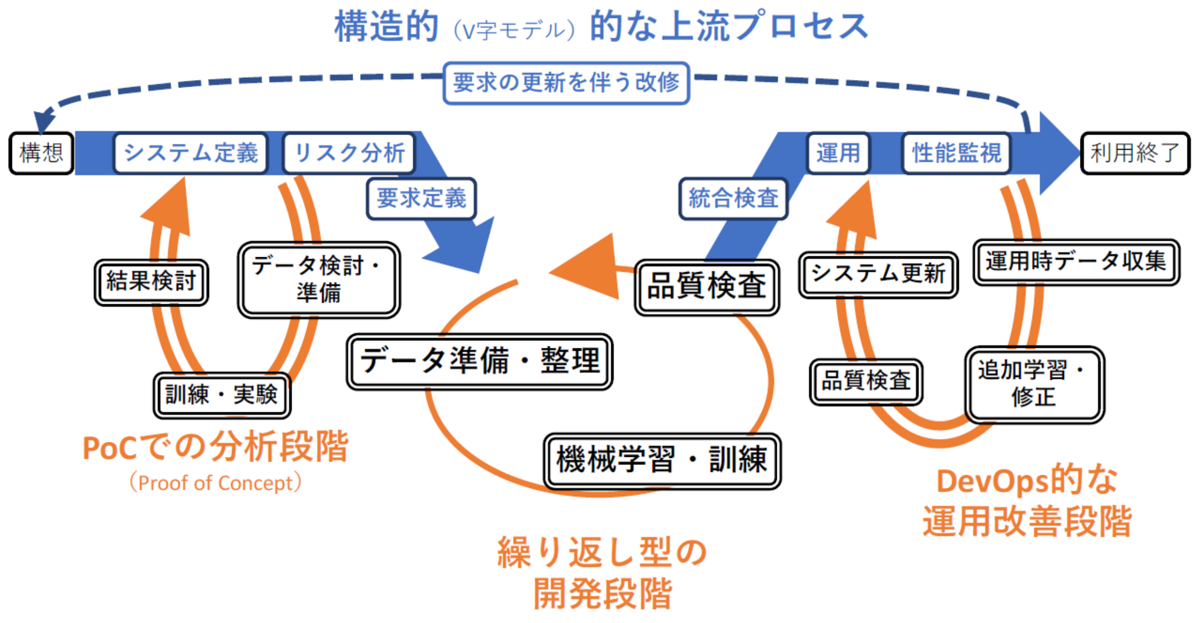
Difference in SE: Deliverables



※ 工程はかなり簡素化
※ 実際のシステムでは両方を統合

Difference in SE: Process

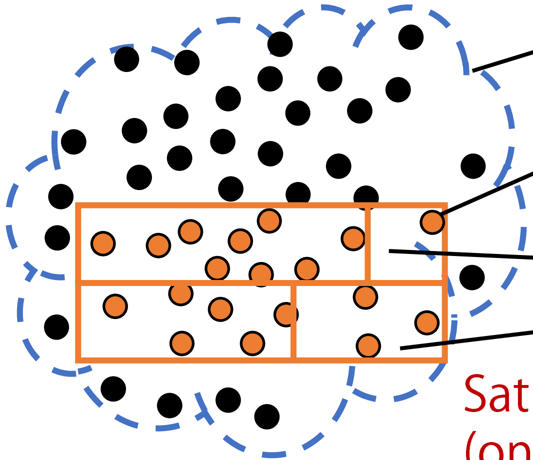
- In addition to the V model
  - PoC (Proof of Concept) is necessary as the feasibility is very uncertain
  - Implementation involves iterations
  - Monitoring and re-training are necessary in operation, e.g., for data drift (changes in the input distribution)



[機械学習品質マネジメントガイドライン 2.1.0] より

Difference in SE: Goals and Requirements Satisfaction

Traditional



Goal: properly judge the health status

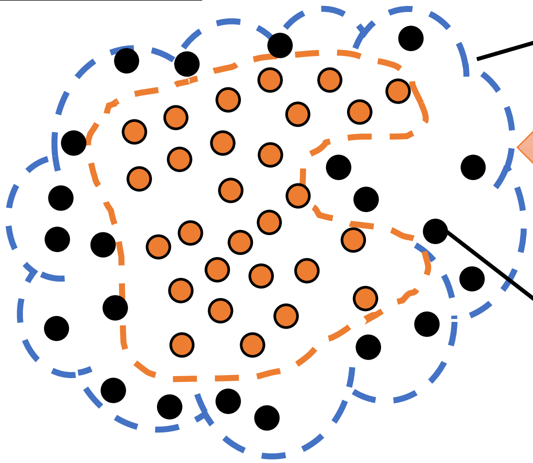
Individual case: for the patient A, we look at XXX values and we should say “need detailed check for diabetes”

Specification as rules

- If HbA1c is more than 6.5, …
- We calculate the area size and then judge a “shadow” in the CT image with this algorithm, …

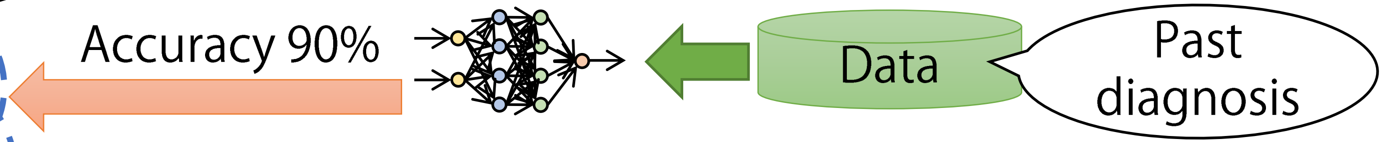
Satisfy perfectly (only) the specified area

ML-based AI



Goal: properly judge the health status

Accuracy 90%



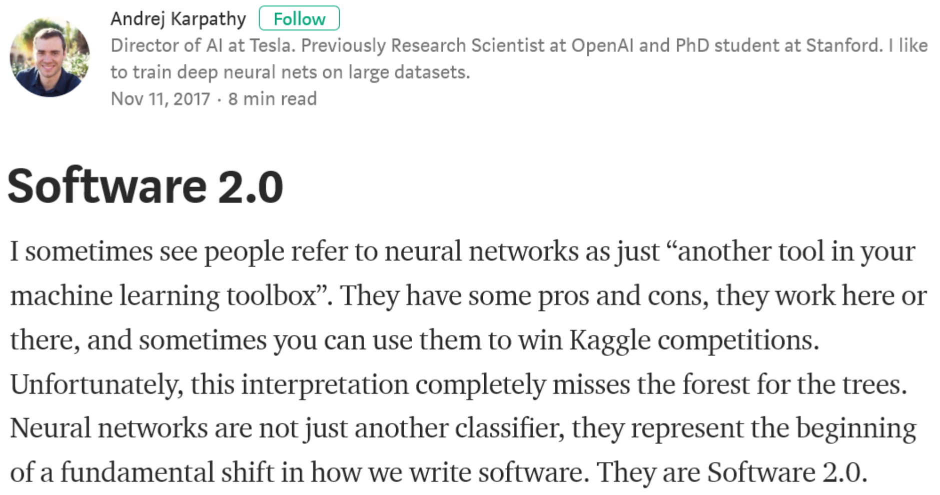
Satisfy most cases (unclear boundary)

Data-driven training

Individual case: may make a mistake even for an obvious case

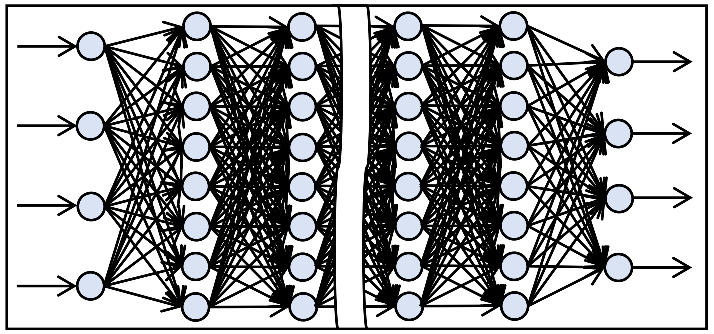
Difference in SE: Implementation

- The delivered software is obtained by searching and configuring a lot of parameter values
  - Millions or more in deep learning


Andrej Karpathy Follow
Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.
Nov 11, 2017 · 8 min read

Software 2.0

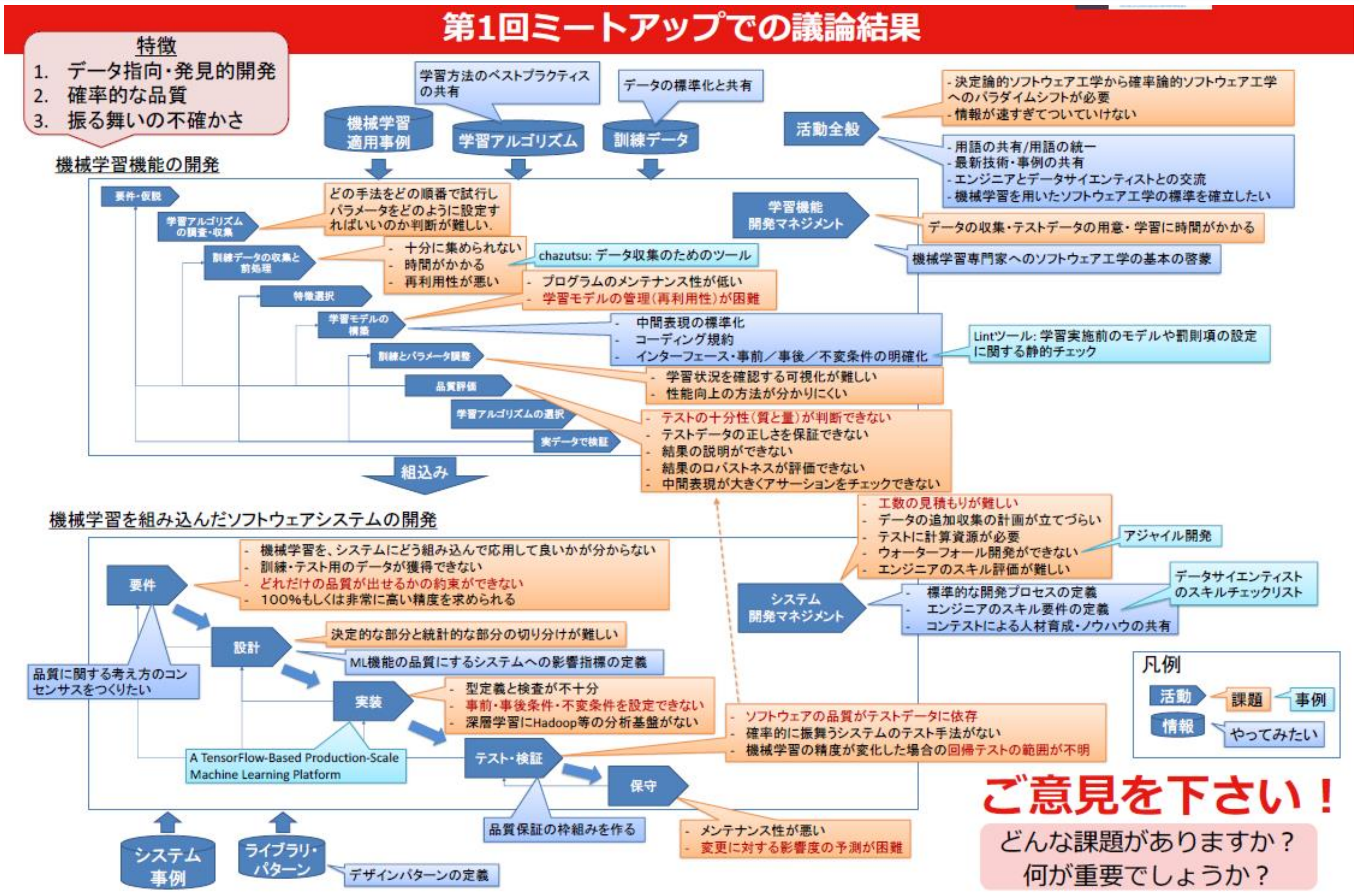
I sometimes see people refer to neural networks as just "another tool in your machine learning toolbox". They have some pros and cons, they work here or there, and sometimes you can use them to win Kaggle competitions. Unfortunately, this interpretation completely misses the forest for the trees. Neural networks are not just another classifier, they represent the beginning of a fundamental shift in how we write software. They are Software 2.0.



[https://medium.com/@karpathy/software-2-0-a64152b37c35]
(access: 2022/07/20)

Challenges of Engineers (2017)

[http://research.nii.ac.jp/~f-ishikawa/work/fose17/]



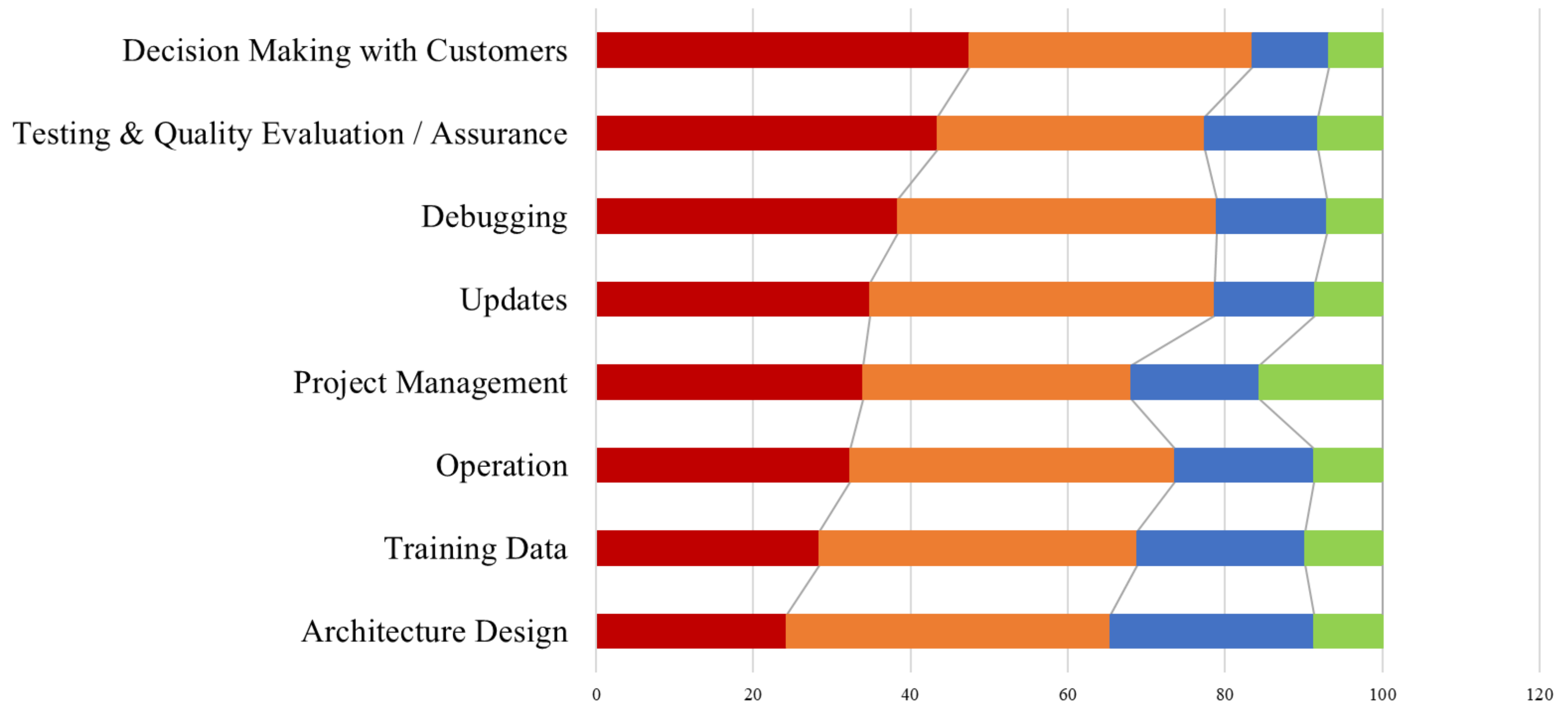
第1回ミートアップでの議論結果
特徴
1. データ指向・発見的開発
2. 確率的な品質
3. 振る舞いの不確かさ
機械学習機能の開発
機械学習適用事例
学習アルゴリズム
訓練データ
学習方法のベストプラクティスの共有
データの標準化と共有
活動全般
- 決定論的ソフトウェア工学から確率論的ソフトウェア工学へのパラダイムシフトが必要
- 情報が速すぎてついていけない
- 用語の共有/用語の統一
- 最新技術・事例の共有
- エンジニアとデータサイエンティストとの交流
- 機械学習を用いたソフトウェア工学の標準を確立したい
要件・仮説
学習アルゴリズムの調査・収集
訓練データの収集と前処理
特徴選択
学習モデルの構築
訓練とパラメータ調整
品質評価
学習アルゴリズムの選択
実データで検証
どの手法をどの順番で試行しパラメータをどのように設定すればいいのか判断が難しい.
- 十分に集められない
- 時間がかかる
- 再利用性が悪い
chazutsu: データ収集のためのツール
- プログラムのメンテナンス性が低い
- 学習モデルの管理(再利用性)が困難
- 中間表現の標準化
- コーディング規約
- インターフェース・事前／事後／不変条件の明確化
Lintツール: 学習実施前のモデルや罰則項の設定に関する静的チェック
- 学習状況を確認する可視化が難しい
- 性能向上の方法が分かりにくい
- テストの十分性(質と量)が判断できない
- テストデータの正しさを保証できない
- 結果の説明ができない
- 結果のロバストネスが評価できない
- 中間表現が大きくアサーションをチェックできない
学習機能開発マネジメント
データの収集・テストデータの用意・学習に時間がかかる
機械学習専門家へのソフトウェア工学の基本の啓蒙
組込み
機械学習を組み込んだソフトウェアシステムの開発
要件
設計
実装
テスト・検証
保守
- 機械学習を、システムにどう組み込んで応用して良いかが分からない
- 訓練・テスト用のデータが獲得できない
- どれだけの品質が出せるかの約束ができない
- 100%もしくは非常に高い精度を求められる
決定的な部分と統計的な部分の切り分けが難しい
ML機能の品質にするシステムへの影響指標の定義
品質に関する考え方のコンセンサスをつくりたい
- 型定義と検査が不十分
- 事前・事後条件・不変条件を設定できない
- 深層学習にHadoop等の分析基盤がない
A TensorFlow-Based Production-Scale Machine Learning Platform
- ソフトウェアの品質がテストデータに依存
- 確率的に振舞うシステムのテスト手法がない
- 機械学習の精度が変化した場合の回帰テストの範囲が不明
品質保証の枠組みを作る
- メンテナンス性が悪い
- 変更に対する影響度の予測が困難
システム事例
ライブラリ・パターン
デザインパターンの定義
- 工数の見積もりが難しい
- データの追加収集の計画が立てづらい
- テストに計算資源が必要
- ウォーターフォール開発ができない
- エンジニアのスキル評価が難しい
アジャイル開発
システム開発マネジメント
- 標準的な開発プロセスの定義
- エンジニアのスキル要件の定義
- コンテストによる人材育成・ノウハウの共有
データサイエンティストのスキルチェックリスト
凡例
活動
課題
事例
情報
やってみたい
ご意見を下さい！
どんな課題がありますか？
何が重要でしょうか？

Challenges of Engineers (2018)

- Questionnaire over around 280 people

We need to use new approaches as the existing ones do not work anymore

We can apply the same approaches but methods, etc., are still immature



We already have dedicated methods, etc.

We can use existing methods, frameworks, or tools

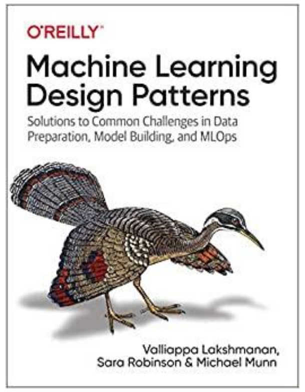
[Ishikawa et al., How do Engineers Perceive Difficulties in Engineering of Machine-Learning Systems? - Questionnaire Survey, CESSER-IP'19]

Typical Issues

- Failing in PoC (said “PoC poverty” or “PoC death” in Japan)
  - High uncertainty and unpredictability
  - Difficulties in achieving high prediction performance, or convincing with incomplete (no 100%) results
  - (Unclear goal setting in the initial phase of the AI trend)
- Different types of technical debts [Sculley et al., Machine Learning: The High-Interest Credit Card of Technical Debt, 2014]
- Unclear testing principles (e.g., no right answer)

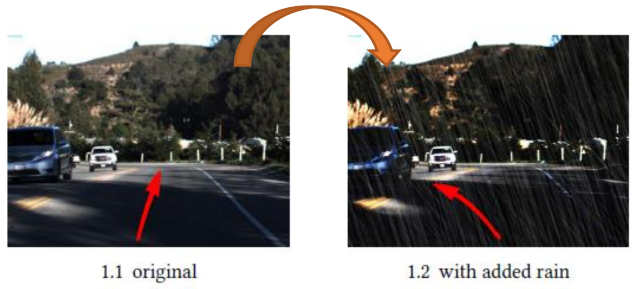
Research and Practice on SE for AI

- A lot of progress after 2018
  - Two quality guidelines in Japan (AIQM, QA4AI)
  - Many testing techniques
  - Dedicated design patterns
  - …


O'REILLY
Machine Learning Design Patterns
Solutions to Common Challenges in Data Preparation, Model Building, and MLOps
Valliappa Lakshmanan, Sara Robinson & Michael Munn

One Example: Whitebox Testing for Deep Neural Network

- Whitebox Testing for Deep Neural Network
  - Generate a test suite by adding noises such as rains by optimization, i.e., search-based
  - 1. Maximize "neuron coverage" to trigger diverse behaviors
  - 2. Maximize "undesirable output changes for input changes"
    - Called metamorphic testing: "if we change the input this way, the output should change in that way"



Input: add rain

Output: angle should not change

[Pei et al., DeepXplore: Automated Whitebox Testing of Deep Learning Systems, 2017]
[Tian et al., DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars, 2018]

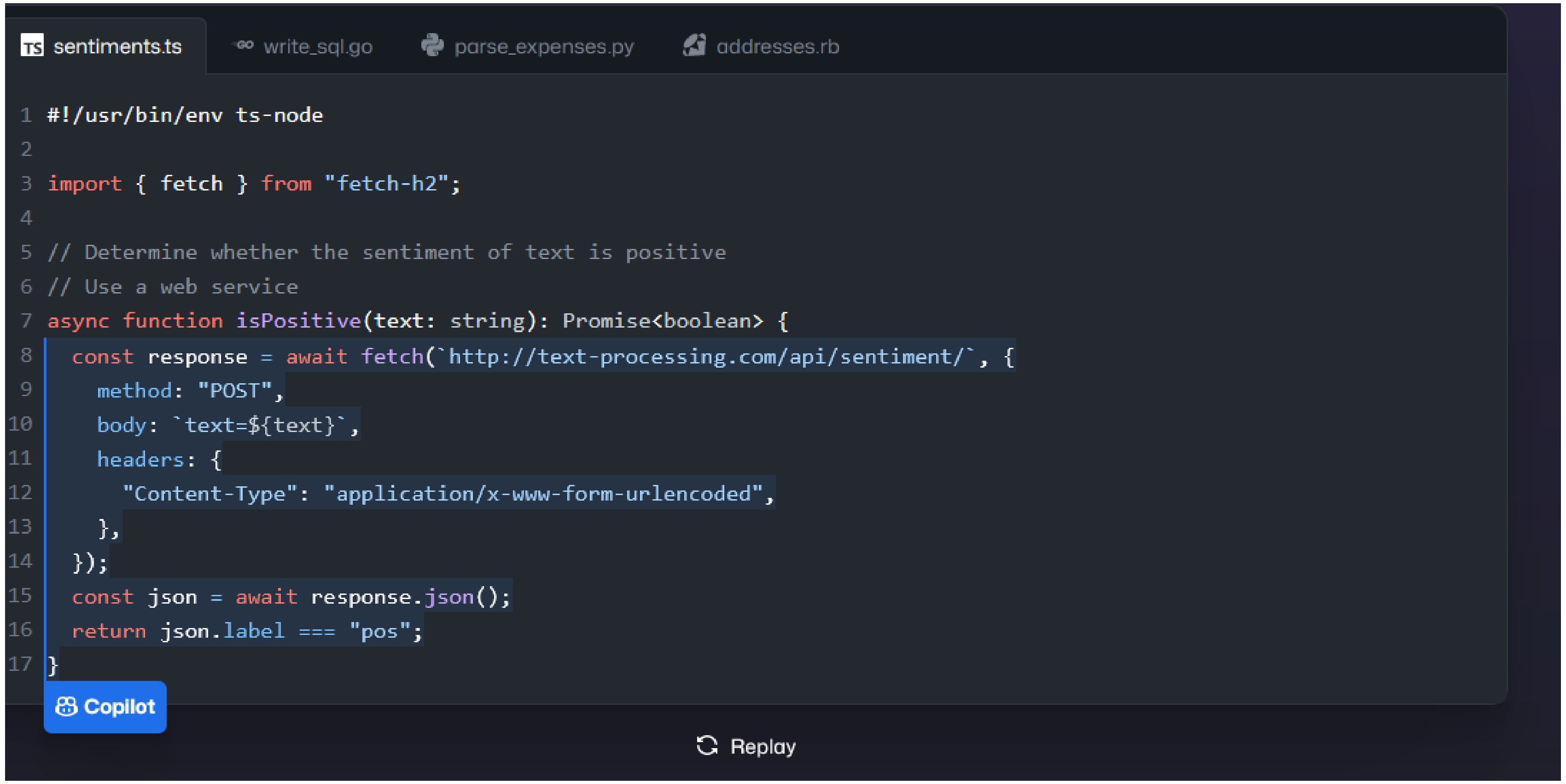
TOC

LLM and Generative Chat AI

- Very active discussion on LLMs (large language models) and generative chat AIs based on them, e.g., ChatGPT
- How should we perceive them?
  - New tools that help software engineering activities? (Generative AI for SE)
  - New way of solution construction that should be helped by software engineering (SE for Generative AI)
    - Instead of "system development", we can "guide by prompts" to obtain a "custom dialogue" as a solution

Generative AI for SE (1)

■Code assistance has been popular (GitHub Copilot)



```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";

// Determine whether the sentiment of text is positive
// Use a web service
async function isPositive(text: string): Promise<boolean> {
  const response = await fetch(`http://text-processing.com/api/sentiment/`, {
    method: "POST",
    body: `text=${text}`,
    headers: {
      "Content-Type": "application/x-www-form-urlencoded",
    },
  });
  const json = await response.json();
  return json.label === "pos";
}
```

[https://github.com/features/copilot/] (2023/6/3 Access)

Generative AI for SE (2)

- Some observed trials in Japan
  - (in blogs, not research papers)
  - Generate functional specification and use case diagrams from a rough description [https://zenn.dev/yamadamadamada/articles/40d0594c1c6375]
  - Discuss design trade-offs [https://dev.classmethod.jp/articles/software_architecture_conversation_with_ai/]
  - Generate and review test cases [https://prtn-life.com/blog/chatgpt-unittest]
  - …

  ()

SE for Generative AI (1)

- Currently, people are rapidly developing the "prompt engineering" principles

[https://www.promptingguide.ai/j] (2023/6/3 access)

```
The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.
A: Adding all the odd numbers (9, 15, 1) gives 25. The answer is False.

The odd numbers in this group add up to an even number: 17,  10, 19, 4, 8, 12, 24.
A: Adding all the odd numbers (17, 19) gives 36. The answer is True.

The odd numbers in this group add up to an even number: 16,  11, 14, 4, 8, 13, 24.
A: Adding all the odd numbers (11, 13) gives 24. The answer is True.

The odd numbers in this group add up to an even number: 17,  9, 10, 12, 13, 4, 2.
A: Adding all the odd numbers (17, 9, 13) gives 39. The answer is False.

The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.
A:
```

Few-shot prompting
+
Chain-of-thought prompting

SE for Generative AI (2)

- Can we further consider "SE-like" activities for (a solution by) generative chat AI?
  - Requirements analysis and feasibility analysis
  - Design and reuse
    - This is some done? (prompt engineering, plug-in, etc.)
  - Testing (checking sufficiency for a certain scope of a goal)
  - Maintenance (tracking updates, etc.)

(this is to be investigated…)

Summary

- Recent and ongoing discussion with specific focus on AI
  - Many attractive techniques have been investigated in research and leveraged in practice though it’s not like “everyone is using”
  - SE for AI has been investigated a lot especially with implementation-level issues with deep learning, e.g., testing
  - ChatGPT is making another impact on SE