HOW DOES MACHINE LEARNING CHANGE SOFTWARE DEVELOPMENT PRACTICES?

Ярослав Голубев

Методы машинного обучения в области программной инженерии

2019

HOW DOES MACHINE LEARNING CHANGE SOFTWARE DEVELOPMENT PRACTICES?

Zhiyuan Wan, Xin Xia, David Lo and Gail C. Murphy

2019, IEEE Transactions on Software Engineering

14 интервью

80 утверждений

342 респондента из 26 стран

RESEARCH QUESTION 1

Как внедрение ML в систему влияет на практики разработки программного обеспечения?

↓ Подготовка ↓ Разработка ↓ Тестирование ↓ Поддержка

RESEARCH QUESTION 2

Как меняются рабочие навыки прикладной психологии?

↓ Разнообразие навыков

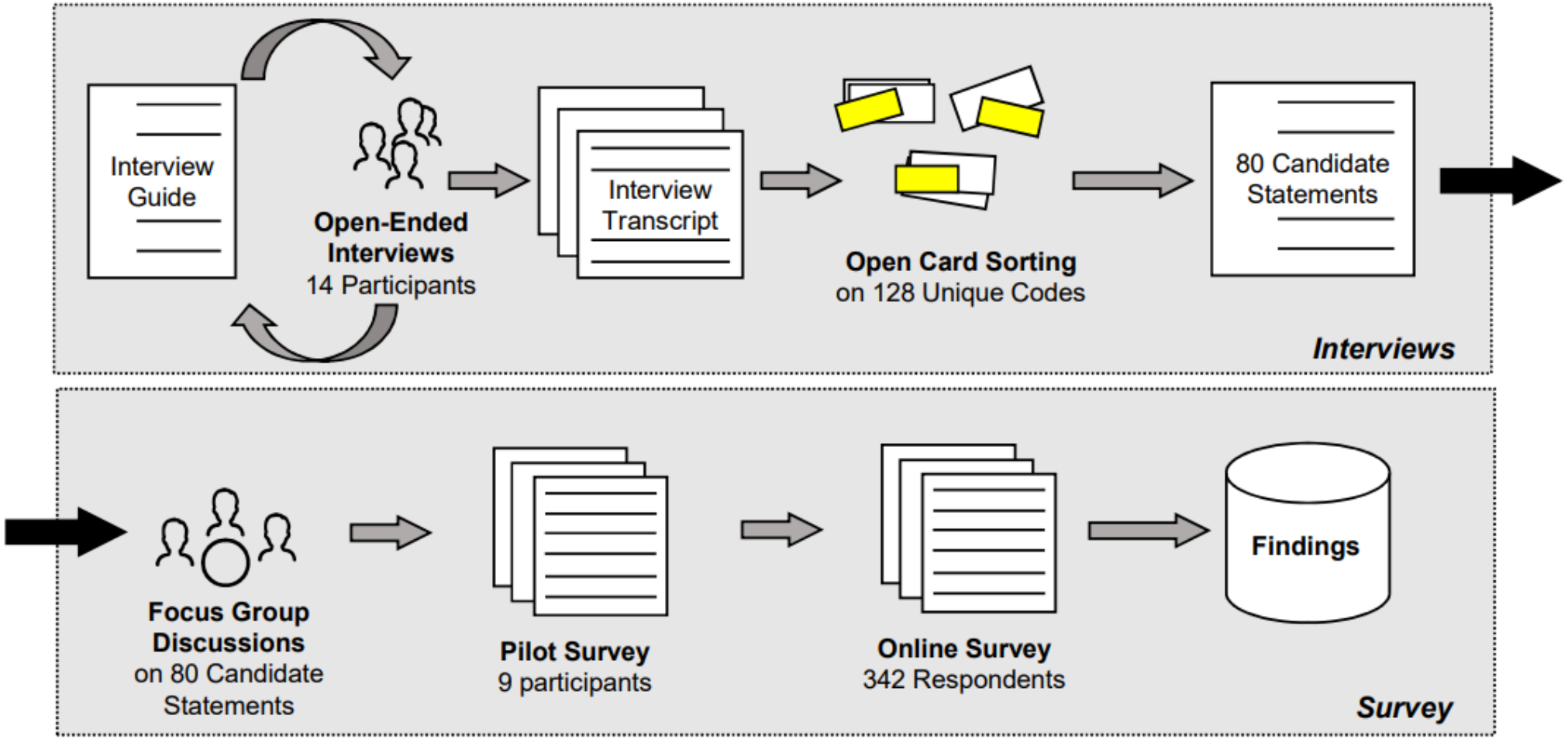
↓ Решение проблем

↓ Планирование работы

↓ Коммуникация

СТАДИИ РАЗРАБОТКИ С ML

МЕТОДОЛОГИЯ



1. ИНТЕРВЬЮ

14 разработчиков, 30–45 минут.

1. Демографические вопросы.
2. Открытый вопрос об отличиях с ML.
3-4. Список нераскрытых тем (два списка, SE и психология).

В среднем, каждый говорил о 6-7 темах.

1. ИНТЕРВЬЮ

SWEBOK: **Requirements** (9 interviewees), SWEBOK: **Design** (6 interviewees), SWEBOK: **Construction** (10 interviewees), SWEBOK: **Tools** (7 interviewees), SWEBOK: **Testing** (9 interviewees), SWEBOK: **Quality** (5 interviewees), SWEBOK: **Maintenance** (4 interviewees), SWEBOK: **Process** (8 interviewees), SWEBOK: **Configuration Management** (3 interviewees).

Work: **Skill Variety** (10 interviewees), Work: **Job Complexity** (5 interviewees), Work: **Problem Solving** (4 interviewees), Work: **Task Identify** (7 interviewees), Work: **Autonomy** (1 interviewees), Work: **Interdependence** (4 interviewees), and Work: **Interaction Outside the Organization** (1 interviewees).

1. ИНТЕРВЬЮ

Роль	ML	Не ML
Разработка	5	6
Архитектура	5	3
Менеджмент	2	2
Тестирование	2	3

2. АНАЛИЗ ДАННЫХ

Интервью были транскрибированы и проанализированы с помощью NVivo. Всего получилось 295 утверждений, который были объединены по смыслу в 128. На последних интервью наступало «насыщение» и новых утверждений не появлялось.

Данные утверждения были независимо категоризированы двумя исследователями с Cohen’s Kappa = 0.78. После чего они были разделены по темам и сформированы в 80 утверждений.

3. ФОКУС ГРУППА

Цель: упросить список вопросов, оставив только самые релевантные.
3 раунда по 2 часа, по 3 человека: профессиональные разработчики из Huawei, Baidu, Alibaba.

7 утверждений убраны, так как все опрошенные посчитали, что разницы нет.

42 утверждения убраны, так как больше половины участников посчитало, что разницы нет.

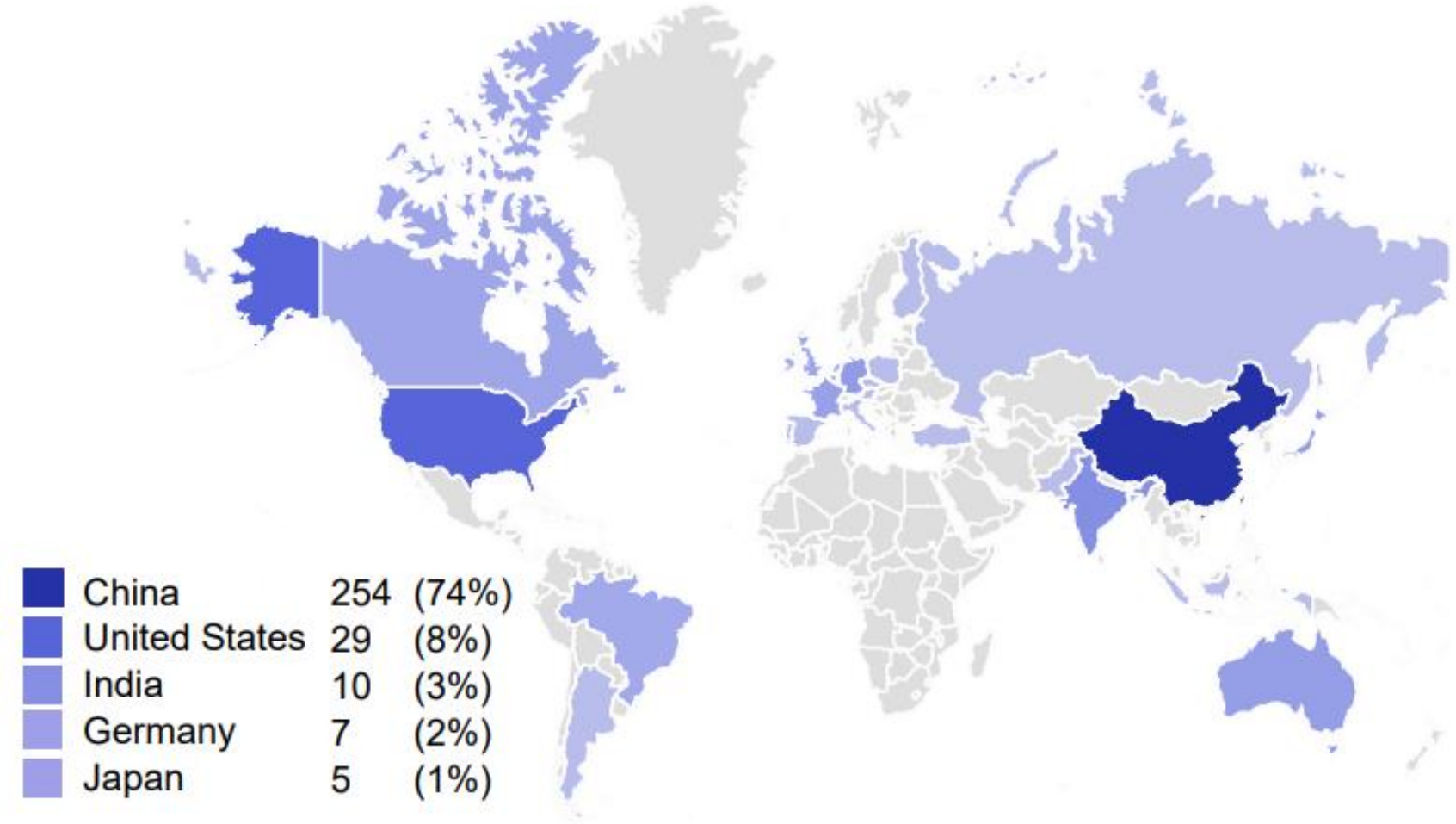
4. ОПРОС

1. Работники Amazon, Alibaba, Baidu, Google, Hengtian, IBM, Intel, IGS, Kodak, Lenovo, Microsoft, Morgan Stanley.
2. 1381 разработчик 18 самых популярных ML репозиториев на гитхабе (TensorFlow, PyTorch). Тут так же разделены те, кто разрабатывает ML фреймворк-тул-библиотеку и ML приложение.

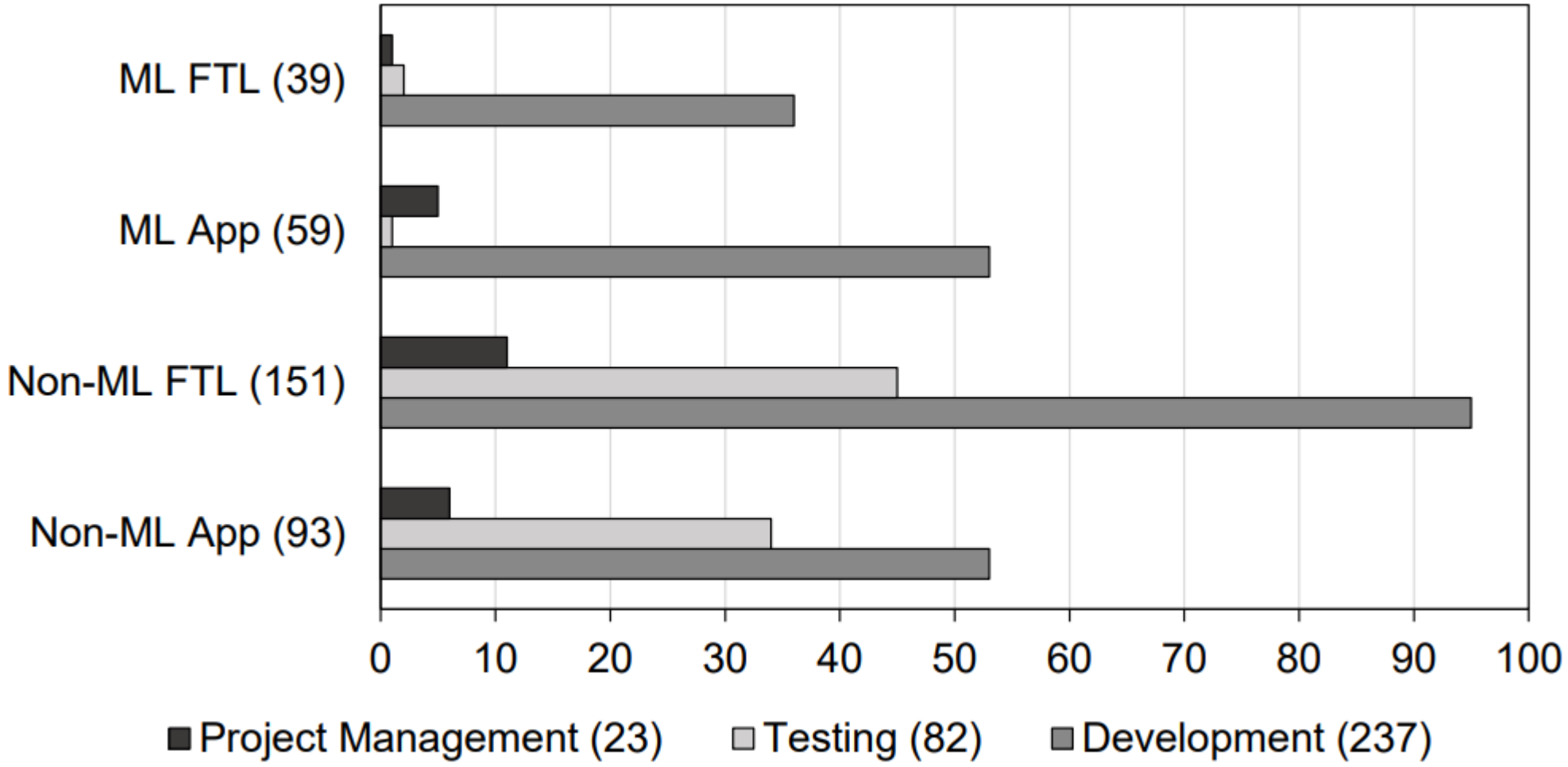
Демография: **«Что лучше всего описывает основную область, в которой вы работаете?»**

Всего **342 ответа**.

4. ОПРОС



4. ОПРОС


ML FTL (39)
ML App (59)
Non-ML FTL (151)
Non-ML App (93)
0
10
20
30
40
50
60
70
80
90
100
Project Management (23)
Testing (82)
Development (237)

РЕЗУЛЬТАТЫ, ВОПРОС 1, ПОДГОТОВКА

1. Подготовка к разработке в ML более неочевидная, «общая». Часто внедрение подается как концепт возможного будущего, а не как набор конкретных точечных улучшений. Если требования и предъявляются, то к количественным характеристикам.
2. **При выборе метода в ML-системах требуется значительно больше предварительных тестов.** Очень часто множество разных алгоритмов подходит к данному типу данных и нужны множественные тесты.
3. **Обязательно следует учитывать предсказываемое ухудшение качества работы ML-систем.**

РЕЗУЛЬТАТЫ, ВОПРОС 1, АРХИТЕКТУРА

4. *Высокоуровневая архитектура ML-систем относительно постоянна.* В то время как для обычных приложений это чаще более творческий процесс.
5. Элементы в ML-системе более связаны друг с другом, чем в большинстве других систем. Ошибка в обработке данных может сделать бессмысленной даже самую эффективную модель.
6. Низкоуровневая архитектура ML-систем, напротив, более разнообразна. При построении моделей возможно сотни и тысячи вариаций.
7. **В связи с этим, разработка архитектуры ML-систем более времязатратна и более итеративная.**

РЕЗУЛЬТАТЫ, ВОПРОС 1, ПРАКТИКИ

8. При разработке ML-систем необходимо писать меньше кода. Чаще это не функционал, а обработка данных, работа с фичами, визуализация, статистика и выбор гиперпараметров.
9. В ML-системах меньше переиспользования кода. Это связано с тем, что они заточены на качество, которое зависит от данных.
10. Дебаггинг в ML-системах направлен не на поиск багов, а на улучшение качества работы.
11. В дебаггинге ML-систем бОльшую роль играет творческий подход. Это связано с огромной свободой настройки модели.
12. В ML-системах баги реже связаны с самим кодом, больше с данными.

РЕЗУЛЬТАТЫ, ВОПРОС 1, ТЕСТИРОВАНИЕ

13. При тестировании ML-систем значительно ниже повторяемость из-за большого количества источников случайности.
14. Тестирование в ML-системах часто означает просто многократный прогон и усреднение данных.
15. *Автоматизированные тестирующие тулы не так часто используются в ML.* (FTL–App разница)
16. **Создание тестовых датасетов очень трудозатратно.** (FTL–App разница)
17. **Хорошие тестовые результаты не гарантируют успешной работы ML-системе в продакшене.**
18. Слишком плохие и слишком хорошие результаты свидетельствуют об ошибках.

РЕЗУЛЬТАТЫ, ВОПРОС 1, ПОДДЕРЖКА

19. Для поддержки ML-систем необходимо меньше усилий. Это связано с тем, что они часто ухудшаются сами по себе и должны быть способны поддерживаться автоматически.
20. Управление ML-системой означает работу с бОльшим количеством контента. Часто кроме самого кода нужно управлять датасетами, параметрами, гиперпараметрами, и т. д.

РЕЗУЛЬТАТЫ, ВОПРОС 1, ОБЩЕЕ

21. **Эффект сильно зависит от конкретных данных, ML — не общее решение.** (FTL–App разница)
22. **Предобработка данных является одним из основных залогов успеха ML-системы.** Кроме того, это относится и к визуализации данных, фичей.
23. Значительным отличием от разработки традиционного программного обеспечения является тот факт, что для ML-систем нет общих гайдов, инструкций. Сама разработка есть по сути итерационная оптимизация без «правильного ответа» в конце.

РЕЗУЛЬТАТЫ, ВОПРОС 2, НАВЫКИ

24. **Разработка ML-систем требует отдельных навыков в областях «математики, теории информации и статистики».**
25. В целом, разработка ML-систем требует более широкого спектра навыков. Так, «data science» здесь требует не только эффективно собирать и преобразовывать данные, но и чистить и исследовать их. Отдельной сложностью является количество данных.

РЕЗУЛЬТАТЫ, ВОПРОС 2, ПРОБЛЕМЫ

26. Сложности в разработке традиционного программного обеспечения обусловлены в основном архитектурой и разбитием на модули.
27. В то же время, сложность разработки ML-систем лежит в работе с данными.
28. **В традиционных приложениях чаще всего существует прямая и понятная (не обязательно простая) дорога к «хорошей архитектуре», улучшать проект можно шаг за шагом.** В ML-системах же, нет понятия «хорошо» или «плохо» написанная модель. Необходимо экспериментировать, работа нелинейна.

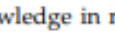
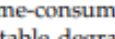
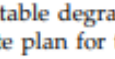
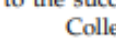
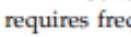
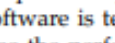
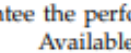
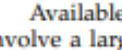
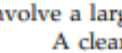
РЕЗУЛЬТАТЫ, ВОПРОС 2, ПЛАНИРОВАНИЕ

29. **В ML-разработке гораздо сложнее строить конкретные планы.** Это связано с тем, что несмотря на то, что архитектура в ML точная и конкретная, она взаимосвязана и сложно-делима. В традиционном программном обеспечении логические модули более дискретны.
30. Отдельной проблемой является то, что непосредственный прирост качества («результат») меньше зависят от разработчика.

РЕЗУЛЬТАТЫ, ВОПРОС 2, ПЛАНИРОВАНИЕ

31. **При разработки ML-систем встречается значительно больше проблем при общении с пользователем.** Во-первых, сложно объяснить нелинейный процесс работы, во-вторых, сложно ответить, «почему А работает, а Б нет».

ОБЩИЕ РЕЗУЛЬТАТЫ

Statement		Likert Distributions				Cliff's Delta		P-values	
		ML (98)	Non-ML (244)	ML FTL (39)	ML App (59)	ML vs. Non-ML	ML FTL vs. ML App	ML vs. Non-ML	ML FTL vs. ML App
Developing my software requires knowledge in math, information theory and statistics.	S24					0.45	-0.19	✔ .000	.320
Detailed design is time-consuming and conducted in an iterative way.	S7					0.32	0.18	✔ .000	.271
Requirements should consider predictable degradation in the performance of software.	S3					0.29	0.11	✔ .000	.433
It is easy to make an accurate plan for the development tasks of my software.	S29					-0.32	0.03	✔ .000	.779
Data processing is important to the success of the whole development process.	S22					0.26	-0.20	✔ .000	.271
Collecting testing dataset is labor intensive.	S15					0.27	-0.26	✔ .000	.188
Developing my software requires frequent communications with the clients.	S31					-0.29	-0.14	✔ .000	.577
My software is tested by using automated testing tools.	S18					0.26	0.48	✘ .000	✔ .001
Good testing results can guarantee the performance of my software in production.	S17					-0.23	0.09	✔ .001	.482
Available data limit the capability my software.	S21					0.22	-0.48	✔ .001	✔ .001
Collecting requirements involve a large number of preliminary experiments.	S2					0.20	0.09	✔ .002	.577
A clear roadmap exists to build my software.	S28					-0.24	0.07	✔ .002	.661
High level architectural design is relatively fixed.	S4					-0.20	0.07	✘ .017	.719
Creativity is important during debugging.	S11					0.12	0.07	.064	.719
My team puts a lot of effort into maintenance of my software.	S19					-0.15	0.21	.065	.188
The higher the performance measures are, the better my software is.	S16					0.08	0.19	.068	.271
Architecture design is complicated for my software.	S26					0.10	0.32	.069	✔ .047
Data modeling is complicated for my software.	S27					0.07	-0.15	.077	.471
Detailed design is flexible.	S6					0.08	0.11	.107	.459
Requirements of my software are uncertain.	S1					0.10	0.11	.151	.482
Testing involves multiple runs of my software to gather a population of quantitative measures.	S14					0.07	0.06	.152	.719
My coding workload is heavy.	S8					0.07	0.08	.223	.665
I have control over the progress towards the target performance.	S30					0.06	0.02	.265	.756
Code reuse happens frequently across different projects.	S9					0.06	-0.12	.265	.482
Debugging aims to locate and fix bugs in my software.	S10					0.06	-0.01	.358	.943
Creating my software requires a team of people, each with different skills.	S25					0.04	0.09	.540	.482
Testing results of my software are hard to reproduce.	S13					0.04	0.04	.546	.787
Low coupling in the components of my software is important.	S5					-0.01	0.08	.861	.459
Configuration management are mainly for the code.	S20					-0.07	-0.14	.894	.943
Software engineering management lacks practical guidance for my software.	S23					-0.01	-0.20	.894	.482
Bugs in my software usually reside in the code.	S12					-0.01	-0.05	.979	.787

ВЫВОДЫ

1. Подготовка к разработке требует больше предварительных тестов и расчёт на изменение качества работы модели в будущем.
2. Разработка более времязатратна и более репетативна.
3. Тестирование сложнее, а хорошие результат не гарантирует хороший продакшен.
4. Работа с данными играет значительную роль и во многом определяет и ограничивает возможные результаты.
5. Требуются более широкие знания, в том числе математика, теория информации и статистика.
6. Сложнее планировать работу.
7. Реже и сложнее получается коммуникация с пользователем.

СЛЕДСТВИЯ

1. Проникнуться случайностью: случайность в данных, случайность в модели.
2. Работа с данными: лучший сбор и генерация данных, лучшая разметка данных, учёт их изменения.

SOFTWARE ENGINEERING FOR MACHINE LEARNING: A CASE STUDY

Saleema Amershi, Andrew Begel, Christian Bird,
Rob DeLine, Harald Gall, Ece Kamar, Nachi Nagappan,
Besmira Nushi, Tom Zimmermann

2019, ICSE-2019

1. РАБОТА С ДАННЫМИ

1. Данные часто объёмны, зависят от контекста, неравномерна и плохо поддаётся описанию.
2. Одним из основных элементов работы в компании является контроль версий данных, который развит значительно слабее, чем для кода.
3. Предлагается унифицированно кодировать метаданные для датасетов.
4. Предлагается использовать функции для своего рода «diff'а» между датасетами.

2. КАСТОМИЗАЦИЯ И ПЕРЕИСПОЛЬЗОВАНИЕ

1. В отличии от обычного кода, модель нельзя «форкануть» и переиспользовать для себя.
2. Мельчайшие требования изменить формат ввода данных в систему может значительно поменять её архитектуру на низком уровне, что создаёт времязатратную проблему.

3. МОДУЛИРОВАНИЕ

1. Модели нельзя «склеивать» между собой, попытка комбинировать входные данные приводит к необходимости переучивать модель.
2. Из-за большого количества данных и параметров, ML-модели значительно более взаимосвязаны, чем обычный код. В связи с этим, улучшение одного «сегмента» может ухудшить другой нелинейно и непредсказуемо.

ПОКАЗАТЕЛИ УСПЕХА

1. Конкретные, сформулированные цели.
2. Последовательное, планомерное исполнение задач.
3. Тщательная документация процесса.
4. Максимальная автоматизация процесса.
5. Отслеживание и сравнение хода процесса.
6. Постоянное улучшение процесса и его подстраивание под цели.

СПАСИБО ЗА ВНИМАНИЕ