

Software Engineering

(10-2) LLM-based AI for Software Engineering

Sokendai / National Institute of Informatics

Fuyuki Ishikawa / 石川 冬樹

f-ishikawa@nii.ac.jp / @fyufyu

<http://research.nii.ac.jp/~f-ishikawa/>

TOC

- Overview of FM/LLM trends
 - Foundation Models
 - Large Language Models
- LLM for Software Engineering
 - Examples and Discussion: LLM for Testing

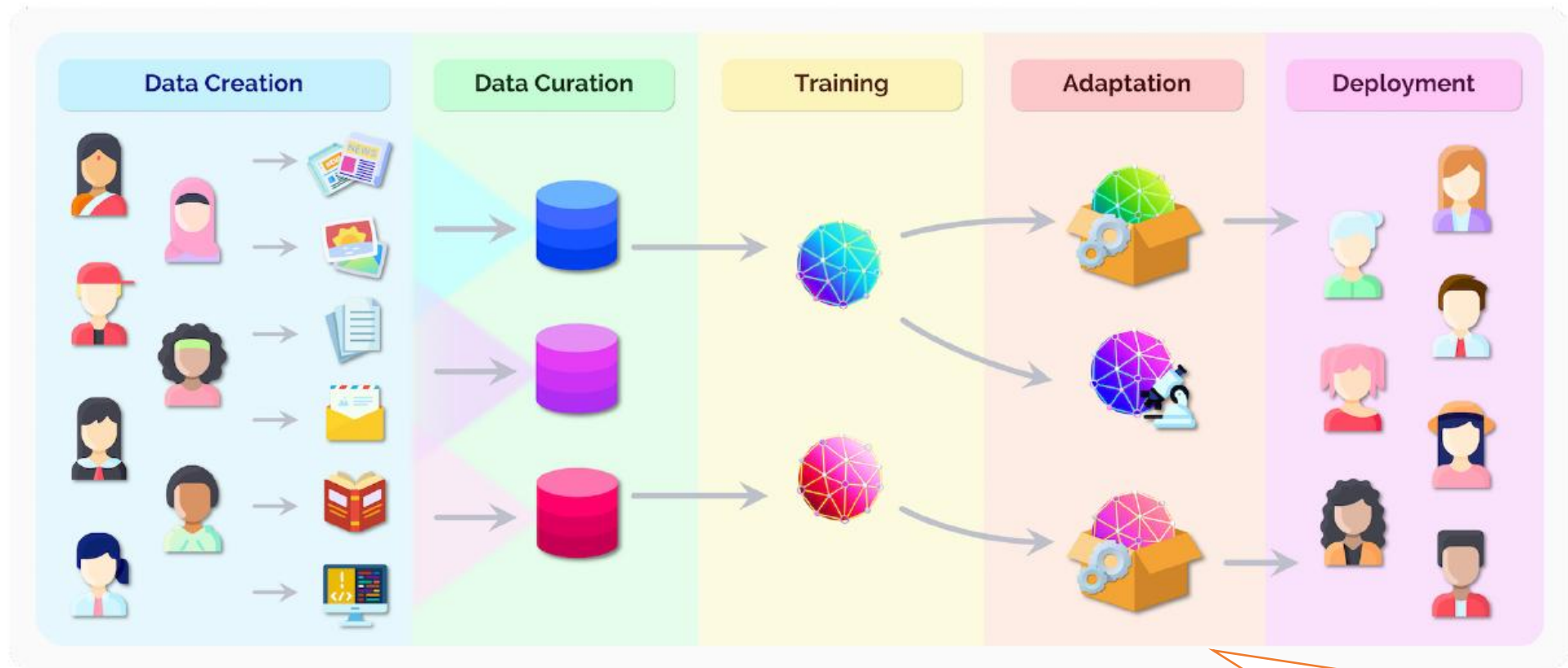
Foundation Models: Definition

■ **Foundation models (FM)**: models trained on broad data at scale and can be adapted to a wide range of downstream tasks

[Bommasani+, On the Opportunities and Risks of Foundation Models, 2021]

- Increasingly popular, especially for natural language processing (NLP) before 2000, not building a model from scratch
- Originally, adaptation was considered by fine-tuning (additional training with task-specific dataset)
- Impressive especially in generative models, e.g., GPT, DALL-E, etc., often with customization by prompting, not fine-tuning

Foundation Models: Social Architecture



Question answering, sentiment analysis, information extraction, image captioning, object recognition, instruction following, ...

Cited from [Bommasani+, On the Opportunities and Risks of Foundation Models, 2021]

Impacts of FM

- Development efficiency

- No need for each organization to collect large amounts of data for training

- Single point of failure

- Undesirable biases or issues of FMs may be widely distributed

- Large cost for building FMs

- Strong dependency on large companies, leading to loss of accessibility and transparency
 - Unclear incentives or imbalanced cost-benefit

Foundation Models: Case of NLP

■ Overview of stories about NLP

- Around 2019, there were active development of FMs such as BERT, GPT2, RoBERTa, T5 and BART
- **Most state-of-the-art models are now based on such FMs** for specific downstream tasks – the competition changed from crafting models from scratch to adapting FMs
- **Large language models (LLMs)**, such as GPT-3 (2020), has shown emergent properties to be customizable broadly by prompts, not training with additional, specialized dataset

Overview of Key Techniques (1)

■ Autoencoder (very original)

[Rumelhard+, Learning Internal Representations by Error Propagation, 1986]

- Encoder: covert raw input to latent representation (compressed and meaningful) by extracting the essence of the input
 - Decoder: convert the latent representation to the original input
- ## ■ Latent representation very useful
- Capturing conceptual similarity and even compositionality, e.g., “Queen=King-Man+Woman”
- ## ■ Encoder, decoder, and encoder-decoder models are very common, e.g., pix2pix translator

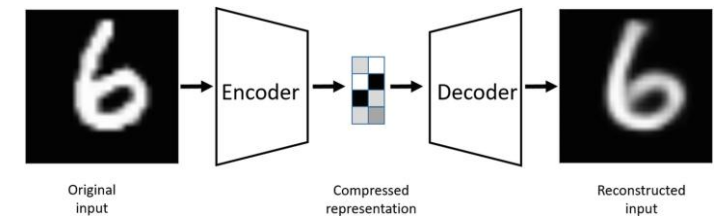


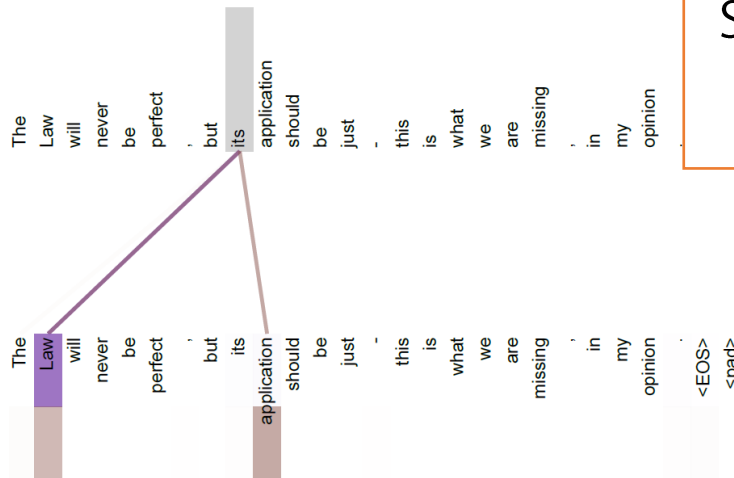
Figure cited from
[Bank+, Autoencoders, 2023]

Overview of Key Techniques (2)

■ Transformer

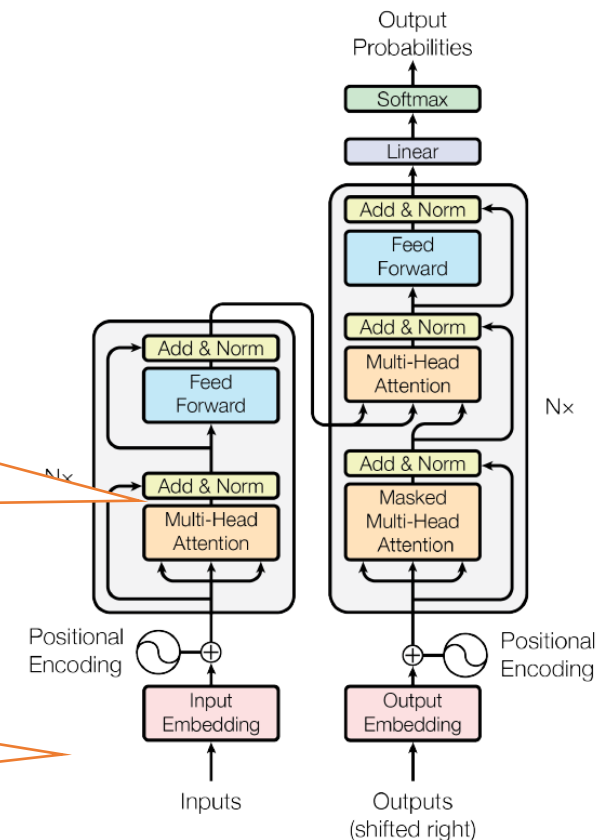
Figures cited from
[Vaswani+, Attention Is All You Need, 2017]

- Did not use convolutional/recurrent networks, leading to higher accuracy and faster training
- Used in many recent NLP models and also in other tasks, e.g., ViT (vision transformer)



Self-attention: the latent representation takes into consideration relationships between words in the same sentence

Encoder-Decoder (left-right)



Overview of Key Techniques (3)

■ BERT

- One of the first FMs for NLP
- Transformer Encoder

Pretraining is done both for binary next-sentence prediction (NSP) and identification of masked words (Mask LM)

NSP:
"the man went to [MASK] store
He bought a gallon [MASK] milk"
-> IsNext = TRUE

Mask LM:
"the man went to [MASK] store
He bought a gallon [MASK] milk"
->
"the man went to the store
He bought a gallon of milk"

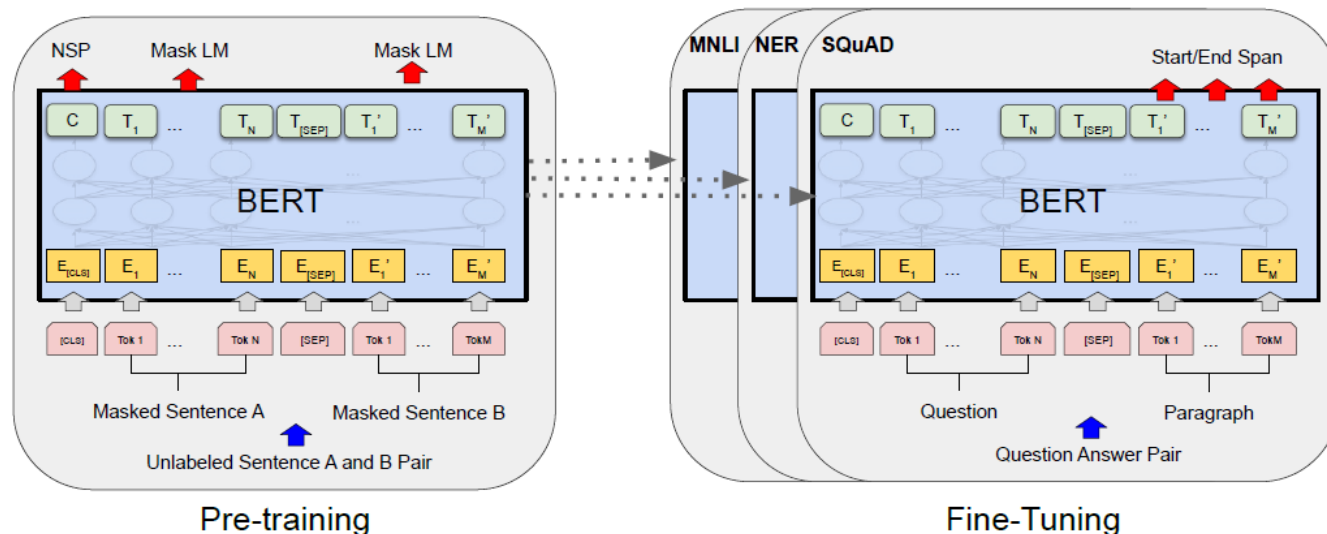


Figure cited from
[Devlin+, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018]

Overview of Key Techniques (4)

■ GPT (initial)

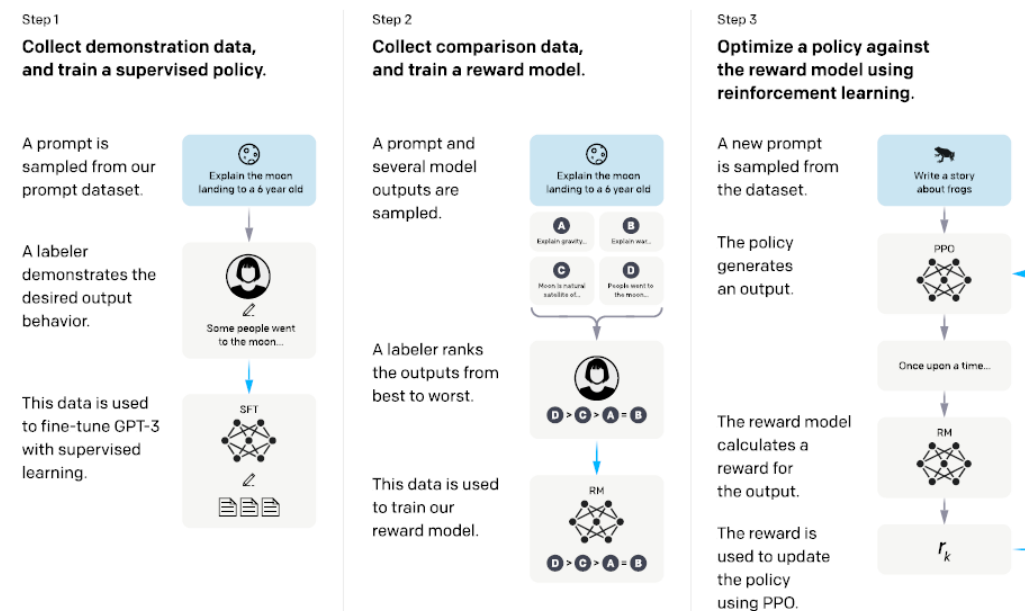
- Originally, an FM intended to be used with fine tuning

- Transformer Decoder [Radford+, Improving Language Understanding by Generative Pre-Training, 2018]

■ InstructGPT

- 3 types of learning for making GPT-3 aligned with human values

→ (Very initial version of) ChatGPT



Cited from [Ouyang+, Training Language Models to Follow Instructions with Human Feedback, 2022]

Foundation Models: Case of Vision

- Overview of stories about computer vision (CV)
 - ImageNet dataset (2009) is one of the initial large datasets for FMs
 - Transfer learning and fine-tuning were common, with the term “pre-trained models” (not “FMs” yet)
 - In response to cost/unavailability of large custom datasets with labels
- Generative models show great performance
 - First with GAN techniques (generative adversarial networks)
 - Recently, diffusion models such as DALL-E (2021), StableDiffusion (2022)
- From pixel-level processing to more commonsense-like one?

Overview of Key Techniques (5)

- GAN (generative adversarial networks)
 - Train two models together: a generative model that captures the data distribution and a discriminative model that distinguishes a generated sample from the training data
 - Led to many impressive generative models such as StyleGAN

[Goodfellow+, Generative Adversarial Nets, 2014]



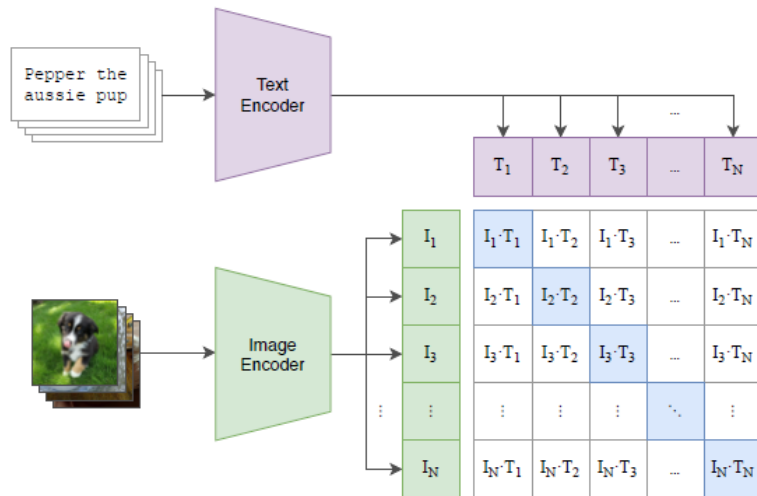
Cited from
[Karras+, A Style-Based Generator Architecture
for Generative Adversarial Networks, 2018]

Overview of Key Techniques (6)

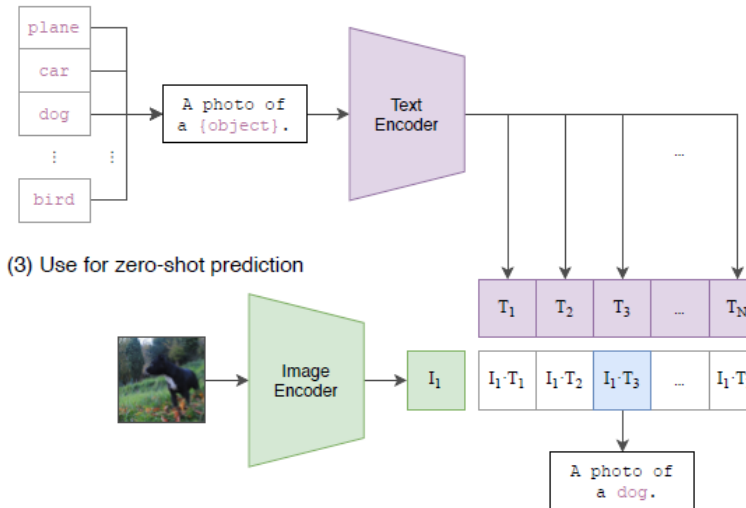
CLIP

- One of the well-known FMs with unsupervised learning (without labels or fixing the target task)
- Learn correlations between texts and images

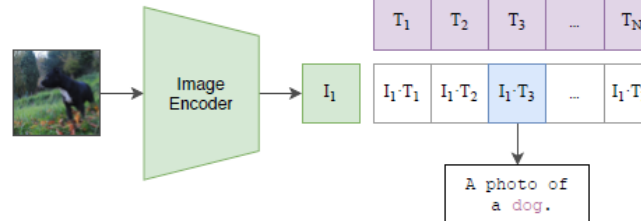
(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



[Radford+, Learning Transferable Visual Models From Natural Language Supervision, 2021]

Overview of Key Techniques (7)

■ Diffusion models

[Sohl-Dickstein+, Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015]

- Very roughly, learn how to restore broken data by iteratively running systematic and slow diffusion steps
- Led to widespread further high-quality models such as DALL-E series and StableDiffusion series
- DALL-E internally uses CLIP to rank the generated image candidates



Figure cited from [Ho+, Denoising Diffusion Probabilistic Models, 2020]

TOC

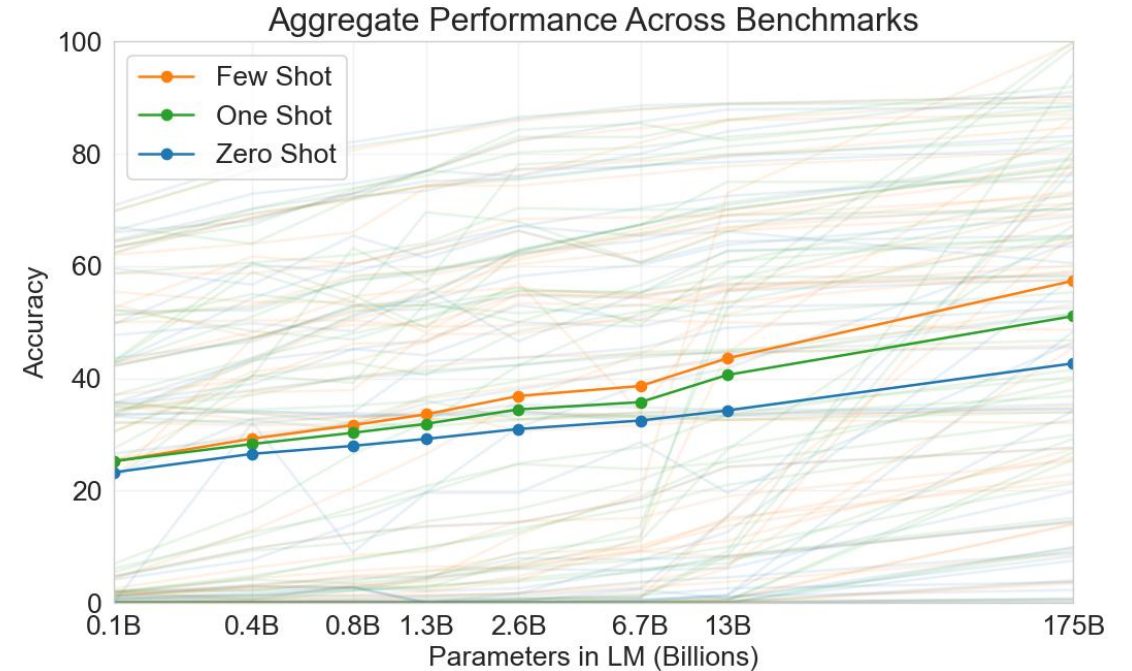
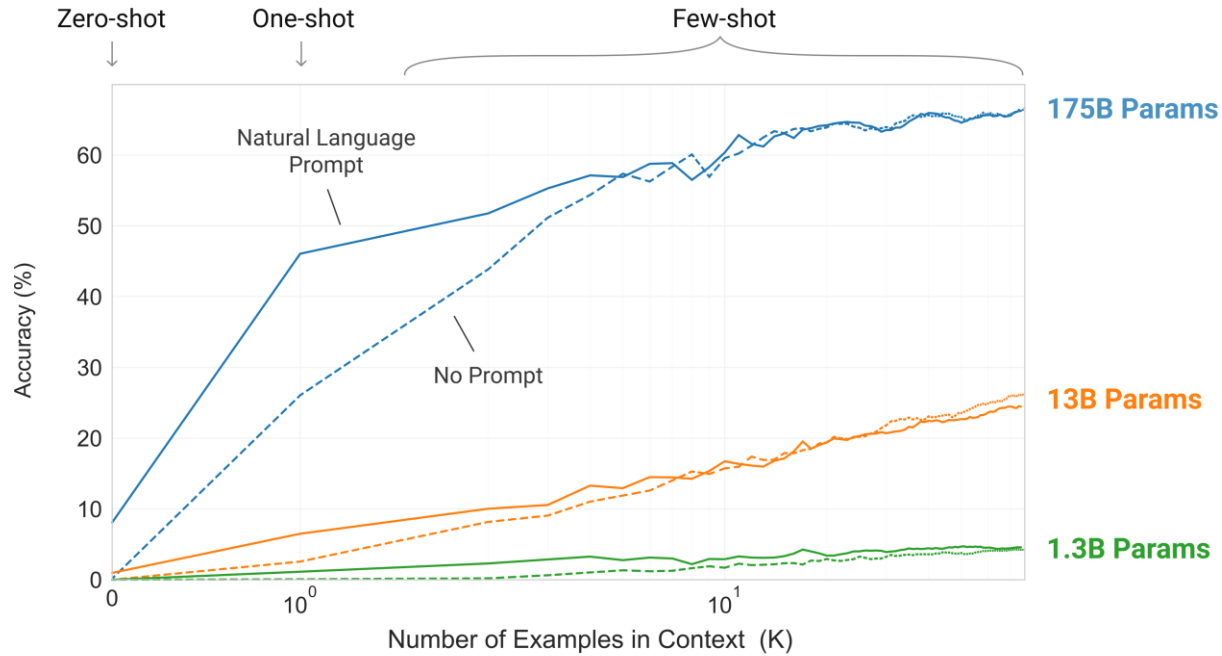
- Overview of FM/LLM trends
 - Foundation Models
 - Large Language Models
- LLM for Software Engineering
 - Examples and Discussion: LLM for Testing

Large Language Models

- Large language models (LLMs) had strong impact as FMs
 - Language models: training to learn probabilities over word sequence
 - “Large”: at the time of GPT-3 (detail was open), 175B parameters were trained with 45TB of compressed plaintext before filtering
 - cf. BERT had 340M (for the “large” version)
- ➡ Led to task-agnostic, few-shot performance, even outperforming fine-tuned, dedicated models

[Brown+, Language Models are Few-Shot Learners, 2020]

Large Language Models: Effect



[Brown+, Language Models are Few-Shot Learners, 2020]

Large Language Models: Toward ChatGPT (Revisited)

■ GPT (initial)

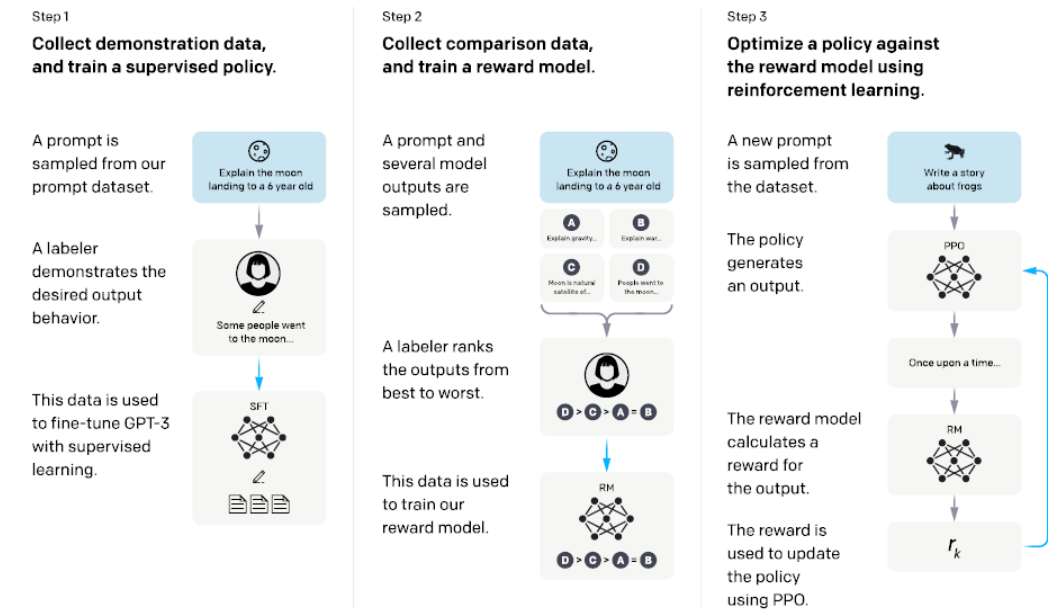
- Originally, an FM intended to be used with fine tuning

- Transformer Decoder [Radford+, Improving Language Understanding by Generative Pre-Training, 2018]

■ InstructGPT

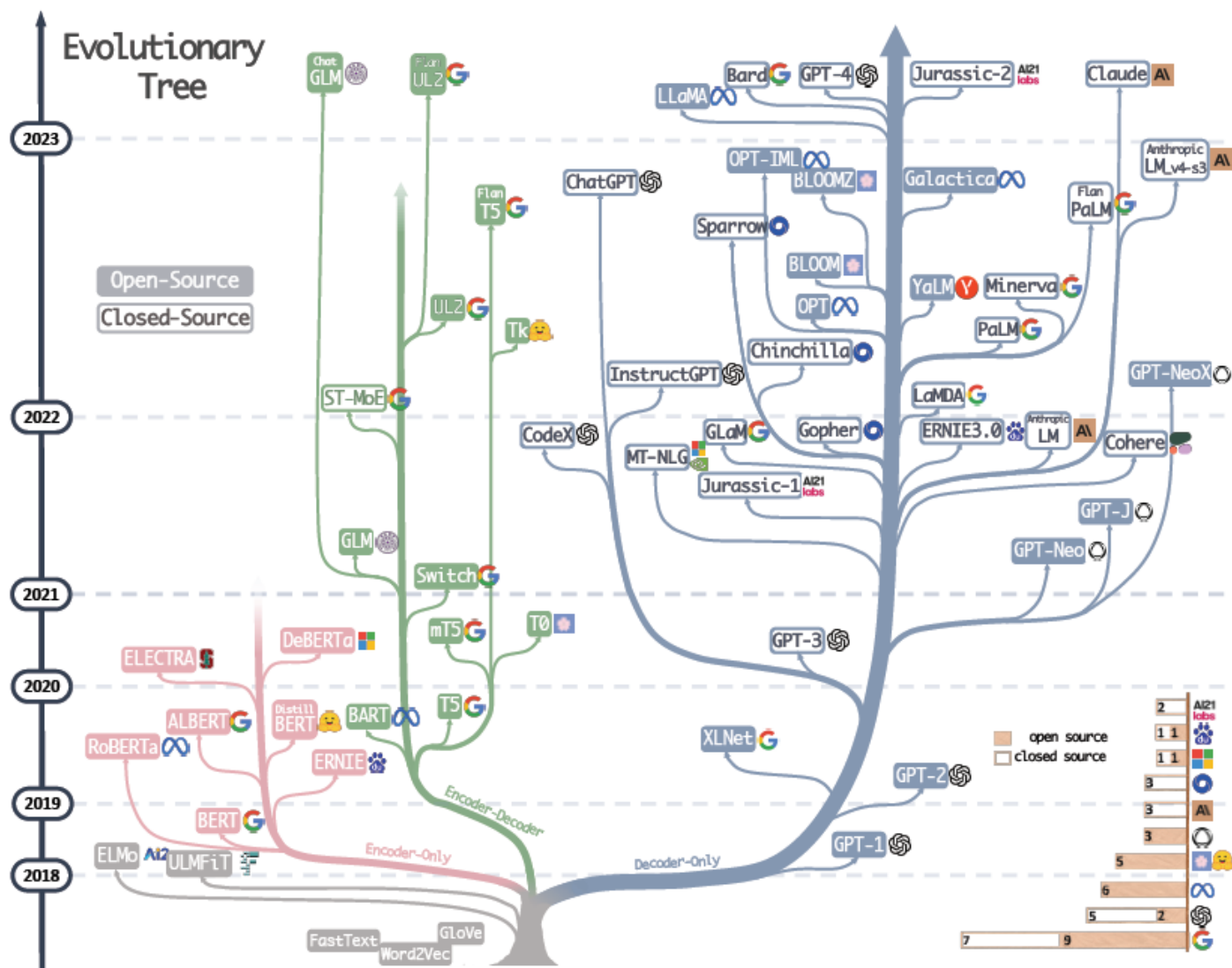
- 3 types of learning for making GPT-3 aligned with human values

→ (Very initial version of) ChatGPT



Cited from [Ouyang+, Training Language Models to Follow Instructions with Human Feedback, 2022]

Large Language Models: Active Effort



Cited from
[Yang+, Harnessing the Power of LLMs in Practice:
A Survey on ChatGPT and Beyond, 2023]

Large Language Models: (Part of) Impact

- ChatGPT got 1,00M active users in 2 months

[<https://www.cnn.com/2023/11/30/chatgpts-one-year-anniversary-how-the-viral-ai-chatbot-has-changed.html>]

- Trillions of dollars in economic growth in the US??

[<https://www.technologyreview.com/2023/03/25/1070275/chatgpt-revolutionize-economy-decide-what-looks-like/>]

- Impact everywhere, including software engineering

- Programming support such as GitHub CoPilot

- More generally, active work in “AI for SE”, different approaches from traditional ones (with building custom models)

[Hou+, Large Language Models for Software Engineering: A Systematic Literature Review, 2023]

Impacts from the Viewpoint of User

- Multi-task, easy use and control
 - High performance in a variety of tasks
 - Zero-Shot/Few-Shot “Customization” only by prompts (at least we can try it before fine-tuning with additional data)
 - Weak barrier to use in natural languages and possibility of iterative and explorative improvement
- ➡ *Without large “development” or “data collection and training”, you can construct your own solution on demand*

Potential Limitations due to Machine Learning Nature

- Not deductive reasoning based on rules/knowledge

- Hallucination issues

- (Relatively) weaker in mathematics and logics

[Frieder+, Mathematical Capabilities of ChatGPT, 2023]

- e.g., it can answer “who is the mother of Tom Cruise”
but not “who is the sun on Mry Lee South”

[<https://www.theguardian.com/technology/article/2024/aug/06/ai-llms>]

- May be affected by training data coverage and biases

- Lack of latest knowledge (without use of web/database)

- Possible biases such as “make a story of nurse” → often “female”

(Very Old) Example of Plausible False Claims



東京特許許可局について教えてください。

それらしい嘘をつく例



東京特許許可局は、日本の特許庁の管轄下にある特許審査・登録の申請受付、審査業務を行う機関の一つです。東京都千代田区にあります。

東京特許許可局では、特許出願や商標出願などの申請を受け付けており、審査官による書類審査や実用新案・意匠の登録審査などを行っています。また、特許出願においては、出願の受付から公開までの手続きも行っています。

東京特許許可局は、日本の特許制度の中心的な機関の一つであり、企業や個人の知的財産権保護に重要な役割を果たしています。



東京特許許可局の最寄り駅はどこですか？



東京特許許可局の最寄り駅は、JR総武線「飯田橋駅」、東京メトロ東西線・有楽町線・南北線「飯田橋駅」、東京メトロ半蔵門線・南北線「永田町駅」、都営地下鉄三田線「半蔵門駅」のいずれかになります。飯田橋駅から徒歩で約10分程度、永田町駅・半蔵門駅からは徒歩で約15分程度の場所にあります。



Regenerate response

駅と路線の対応や距離関係もおかしい

[ChatGPT 2023 Feb 13 ver. (GPT-3.5)]

Default (GPT-3.5)



東京特許許可局について教えてください

お断りの言葉が入るように



申し訳ありませんが、私の知識は2021年9月までのものであり、それ以降の情報は持っていません。したがって、2023年の情報については提供できません。ただし、一般的に東京特許許可局（Tokyo Patent Office）についてお話しします。

東京特許許可局は、日本における特許や実用新案、意匠などの知的財産権に関する申請や登録、審査を担当する機関です。特許許可局は、日本国内の各地に設置されていますが、東京特許許可局はその中でも最も重要な局の一つとして知られています。東京特許許可局の管轄区域は東京都を中心とした広範囲に及びます。

特許許可局は、特許庁（Japan Patent Office, JPO）の下部組織として運営されており、知的財産制度の維持と発展に向けた重要な役割を果たしています。特許許可局は、特許法や実用新案法などに基づいて、新規性や非自明性、産業上の適用可能性などを審査し、特許権の付与を決定します。

具体的な最新情報については、日本特許庁の公式ウェブサイトや関連する公式情報源を確認してください。

[ChatGPT 2023 Jul 20 ver. (GPT-3.5)]

GPT-4



東京特許許可局について教えてください

訓練が進み適切な回答に



「東京特許許可局」は、実際には存在しない架空の組織で、日本語の早口言葉として有名です。このフレーズは、「とうきょうとっきよきよきよきよ」と発音することから、日本語を母国語とする人々でも発音が難しいとされることが多いです。

早口言葉は、言葉の発音を速く連続して行う練習やゲームとして用いられることが一般的で、発音の練習や、言語のリズムと流れを楽しむために使われます。「東京特許許可局」のようなフレーズは、子供から大人まで楽しむことができる、日本語の言語文化の一部です。

[ChatGPT 2023 Jul 20 ver. (GPT-4)]

TOC

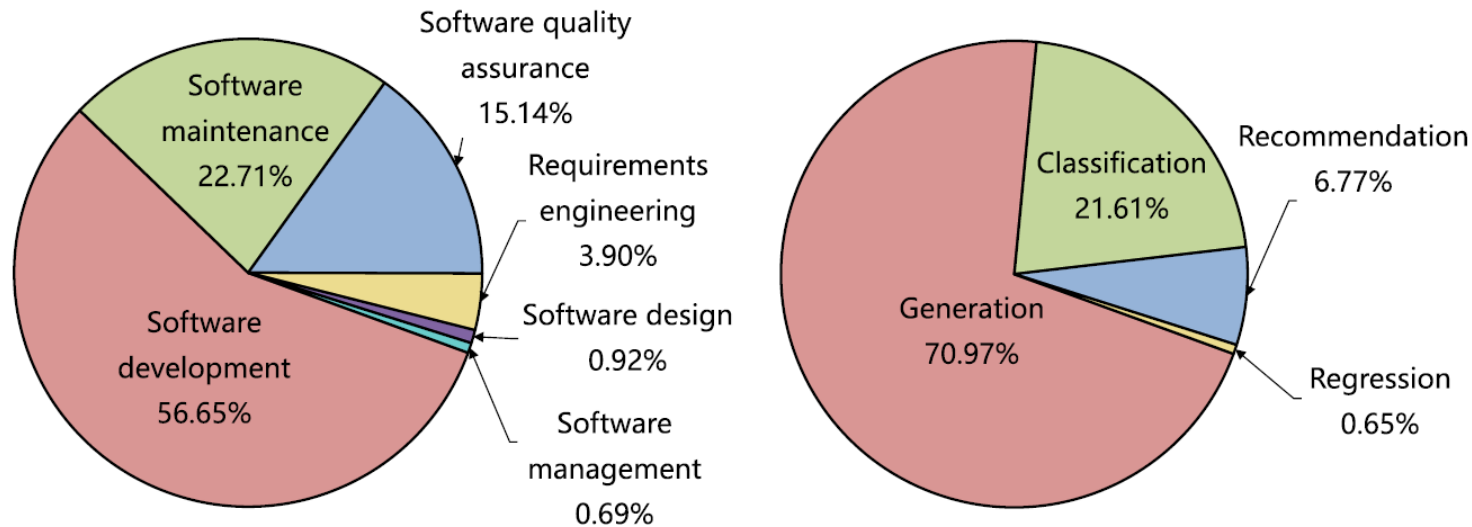
- Overview of FM/LLM trends
 - Foundation Models
 - Large Language Models
- LLM for Software Engineering
 - Examples and Discussion: LLM for Testing

LLM for Software Engineering

- Coding has been one of the key applications of LLM
- Supporting tools/environments
 - GitHub CoPilot, Cursor, Cline, ...
- Google CEO said:
 - Today, more than a quarter of all new code at Google is generated by AI, then reviewed and accepted by engineers. This helps our engineers do more and move faster.
[<https://blog.google/inside-google/message-ceo/alphabet-earnings-q3-2024/#search>]
- Active effort for SE not limited to coding

LLM for Software Engineering: Example of Survey (1)

- “Until early 2024” – that almost means a collection of studies in 2023



(a) Distribution of LLM usages in SE activities. (b) Problem classification based on collected studies.

Fig. 10. Distribution of LLM utilization across different SE activities and problem types.

Cited from [Hou+, Large Language Models for Software Engineering: A Systematic Literature Review, 2024]

LLM for Software Engineering: Example of Survey (2)

■ (続)

Requirements engineering	Anaphoric ambiguity treatment (4) Requirements analysis and evaluation (2) Coreference detection (1) Specification formalization (1) Use cases generation (1)	Requirements classification (4) Specification generation (2) Requirements elicitation (1) Traceability automation (1)
Software design	GUI retrieval (1) Software specification synthesis (1)	Rapid prototyping (1) System design (1)
Software development	Code generation (118) Code summarization (21) Code translation (12) API inference (5) API recommendation (5) Code representation (3) Method name generation (2) Agile story point estimation (1) API documentation smells (1) Data analysis (1) Control flow graph generation (1) Instruction generation (1) Others (14)	Code completion (22) Code search (12) Code understanding (8) Program synthesis (6) Code editing (5) Code comment generation (2) Code recommendation (2) API documentation augment (1) API entity and relation extraction (1) Fuzz driver generation (1) Identifier normalization (1) Type inference (1)

Cited from [Hou+, Large Language Models for Software Engineering: A Systematic Literature Review, 2024]

LLM for Software Engineering: Example of Survey (3)

■ (続)

Software quality assurance	Vulnerability detection (18)	Test generation (17)
	Bug localization (5)	Verification (5)
	Testing automation (4)	Fault localization (3)
	Defect detection (2)	GUI testing (2)
	Static analysis (2)	Binary taint analysis (1)
	Compiler fuzzing (1)	Decompilation (1)
	Invariant prediction (1)	Malicious code localization (1)
	Mobile app crash detection (1)	Resource leak detection (1)
	Test prediction (1)	
Software maintenance	Program repair (35)	Code clone detection (8)
	Code review (7)	Debugging (4)
	Bug reproduction (3)	Review/commit/code classification (3)
	Duplicate bug report detection (3)	Logging (3)
	Log parsing (3)	Code revision (2)
	Sentiment analysis (3)	Vulnerability repair (2)
	API misuses repair (1)	Bug prediction (1)
	Bug triage (1)	Code coverage prediction (1)
	Code review explained (1)	Code-Review defects repair (1)
	Crash bug repair (1)	Dockerfile Repair (1)
	Incivility detection (1)	Patch correctness prediction (1)
	Patch detection (1)	Program merge conflicts repair (1)
	Rename Refactoring (1)	Tag recommendation (1)
	Technical debt payback (1)	Traceability recovery (1)
	Web test repair (1)	Type error repair (1)
	Others (5)	
Software management	Effort estimation (2)	Software tool configuration (1)

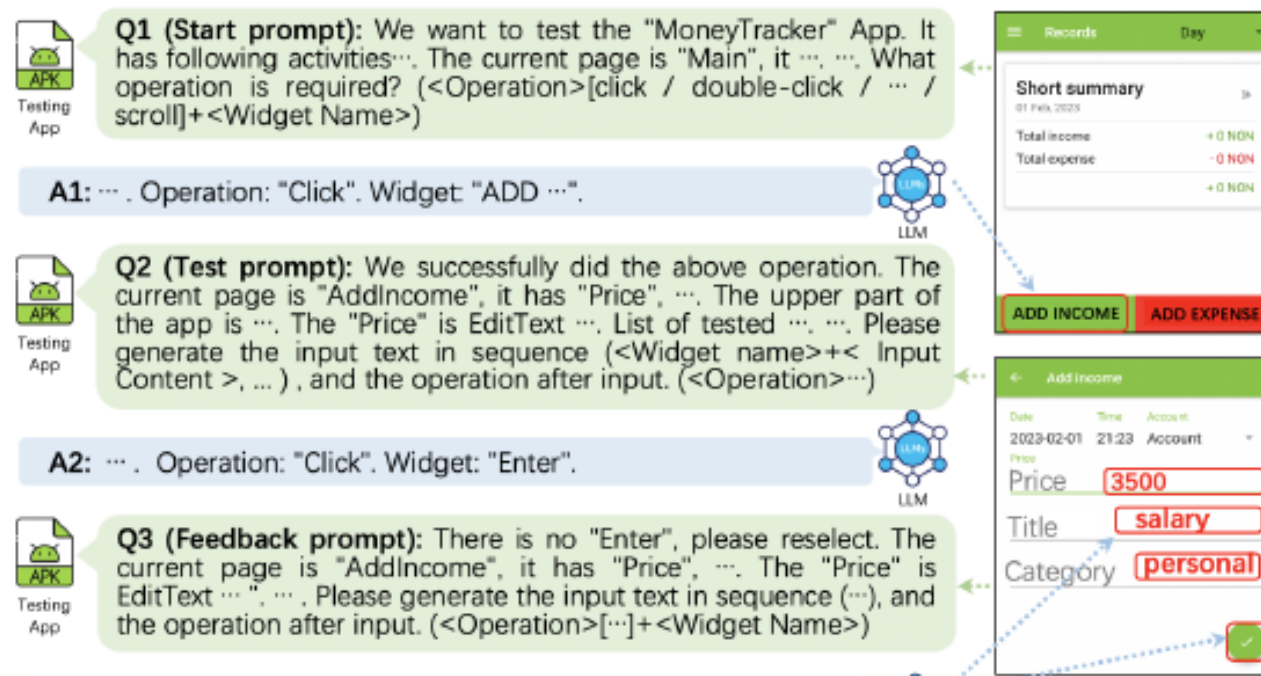
TOC

- Overview of FM/LLM trends
 - Foundation Models
 - Large Language Models
- LLM for Software Engineering
 - Examples and Discussion: LLM for Testing

Example of Research on LLM for Testing (1)

■ Automated explorative testing

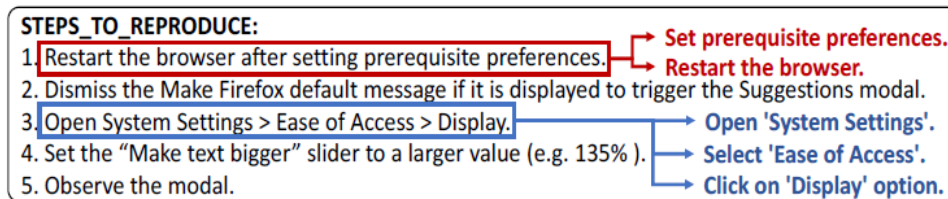
- LLM iteratively suggests the next test action given the current screen and the summary of history tests



[Liu+, Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions, ICSE'24]

Example of Research on LLM for Testing (2)

- LLM makes more structured description of steps to reproduce failures as a knowledge graph
 - Then, other techniques can be applied more easily, such as statistical analysis and automated test generation



(a) To-Be-Processed S2Rs

Please split STEPS_TO_REPRODUCE into steps, especially splitting the step with multiple UI operations into steps with one UI operation.

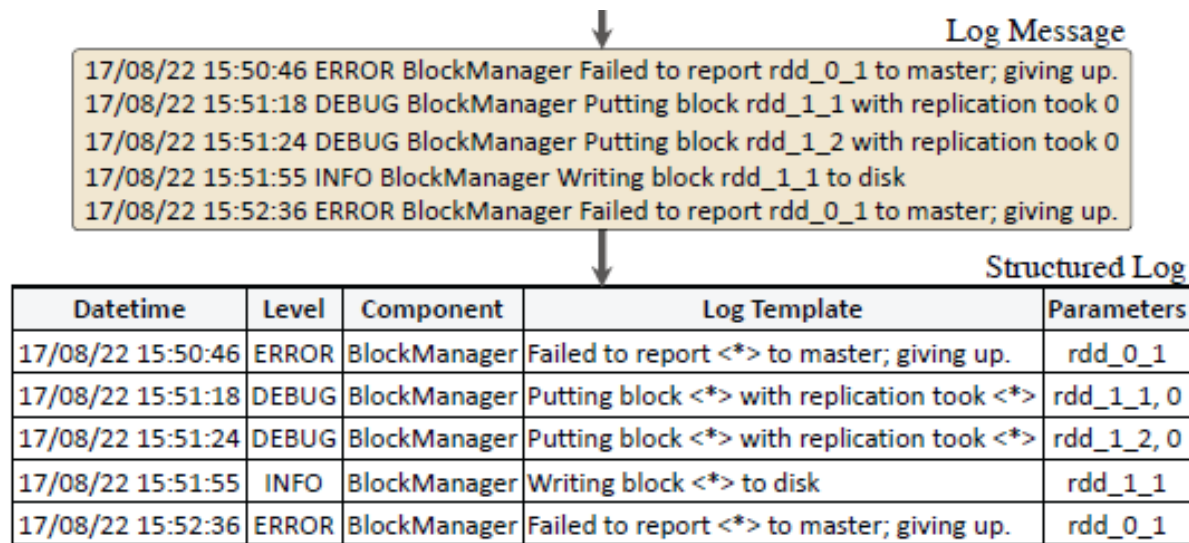
(b) Step Splitting Query

```
[{"STEP": "Set prerequisite preferences.", "STEP_TYPE": "OPERATION"},  
{"STEP": "Restart the browser.", "STEP_TYPE": "OPERATION"},  
{"STEP": "Dismiss the Make Firefox ... the Suggestions modal.", "STEP_TYPE": "OPERATION"},  
{"STEP": "Open 'System Settings'.", "STEP_TYPE": "OPERATION"},  
{"STEP": "Select 'Ease of Access'.", "STEP_TYPE": "OPERATION"},  
{"STEP": "Click on 'Display' option.", "STEP_TYPE": "OPERATION"},  
{"STEP": "Set the 'Make text bigger' ... larger value (e.g. 135%).", "STEP_TYPE": "OPERATION"},  
{"STEP": "Observe the modal.", "STEP_TYPE": "NON_OPERATION"}]
```

[Su+, Enhancing Exploratory Testing by Large Language Model and Knowledge Graph, ICSE'24]

Example of Research on LLM for Testing (3)

- LLM makes structure of logs
 - Logs come from different libraries and have diverse formats
 - A lot of studies on this direction
 - Again, other techniques can be used once the data is structured



[Le+, Log Parsing with Prompt-based Few-shot Learning, ICSE'23]

Features of LLM-based Support

- Supports any types of tasks (at least look so)
 - But we may not use “only LLM” or “end-to-end LLM”
- Accepts unstructured or incomplete inputs
 - Less requirements on data formatting, scheme standardization, preliminary training with large datasets
 - No worry about different representations in natural languages
 - Possibility to complement implicit knowledge (commonsense)
 - Connection to existing techniques that require structured inputs

Review of Traditional Techniques (1) Algorithm/Reasoning

- Task-specific algorithms

- e.g., generating test suites that satisfy All-Pair constraints

- Model-based testing

- e.g., generating a test suite from UML design models

- Symbolic Reasoning / Concolic Testing

- e.g., generating high-coverage test cases with logical solvers

➡ Traditional techniques (enhanced with LLM?) can be stronger if we need to satisfy constraints or improve metrics

Review of Traditional Techniques (2) “AI”

- Supervised learning with task-specific datasets
 - e.g., extracting GUI components from a screenshot
 - Search/optimization methods such as evolutionary computation or reinforcement learning
 - e.g., generating a small test suite with high coverage
 - e.g., automated testing smartphone apps
- ➡ Traditional techniques (enhanced with LLM?) can be stronger if we need to improve quantitative metrics

Discussion: How to Proceed?

- It is very easy to “start” by exploring prompts and input/output formats
 - It seems now the most significant thing is “try” or “fail fast (resolve uncertainty)” rather than “plan carefully”
 - It is easy to just say “human validate the outputs”
- ➡ Is it possible to iterate the cycles of evaluation and improvement? (critical issue in SE for LLM)
 - For example, can you prepare and iterate (ideally automated) “tests for ‘LLM/AI for tests’”??

Summary: LLM/AI for Software Engineering

- Breakthrough by LLM (and foundation models in general)
 - Large impact in software engineering
 - Active effort on coding, almost at the level of practical use (at least in early adapters and some domains)
 - Also support of various tasks: model generation, model validation, test design, test coding, debugging, ...
 - Still evolving a lot