agent | ˈeɪdʒ(ə)nt |

noun

1 a person who acts on behalf of another person or group: *in the event of illness, a durable power of attorney enabled her nephew to act as her agent.*
- a person who manages business, financial, or contractual matters for an actor, performer, writer, etc.: *his agent was able to negotiate a long-term contract.*
- a person or company that provides a particular service, typically one that involves organizing transactions between two other parties: *speak to your letting agent about refurbishing the property.*
- a person who works secretly to obtain information for a government or other official body: *a trained intelligence agent.*

2 a person or thing that takes an active role or produces a specified effect: *these teachers view themselves as agents of social change.*
- a substance that brings about a chemical or physical effect or causes a chemical reaction: *there is an urgent need for new antimicrobial agents to combat infections | the bleaching agent used is hydrogen peroxide.*
- *Grammar* the doer of an action, typically expressed as the subject of an active verb or in a **by** phrase with a passive verb.
- *Computing* an independently operating program, typically one that performs background tasks such as information retrieval or processing on behalf of a user or other program.

AI Agent

- Autonomy: given a task, it can plan the actions for completing the task and make decisions accordingly, on its own.
- Proficiency: it can find the appropriate tools and use them to complete the given task.
- LLMs can be more than chatbots.
  - In fact, many chatbots are internally complicated agent systems, I suspect. But more about this later.

Why build agents?

- “If we wait a bit, maybe LLMs will be even smarter, achieve AGI, and we won’t need complicated agents… no?”
- This is a big question, and not just a rhetorical one 🤓 But here are my personal, biased answers for now.
  - Will AGI be possible based on the Transformer architecture?
  - What are we going to do about the lack of memory?
  - Context is now much longer than before, but it is still finite.
  - Some things are just easier when handled symbolically.

Why work on agents?

- If what we said in the previous slide is true, then we are looking at the beginning of a new form of software system
  - Truly neuro-symbolic: LLMs and traditional programming language making decisions together
  - Intelligent and multi-modal, therefore directly human facing with free-form inputs
  - We’re living in a really exciting time, whether you believe in AGI or not :)

A few of examples (from my own research)

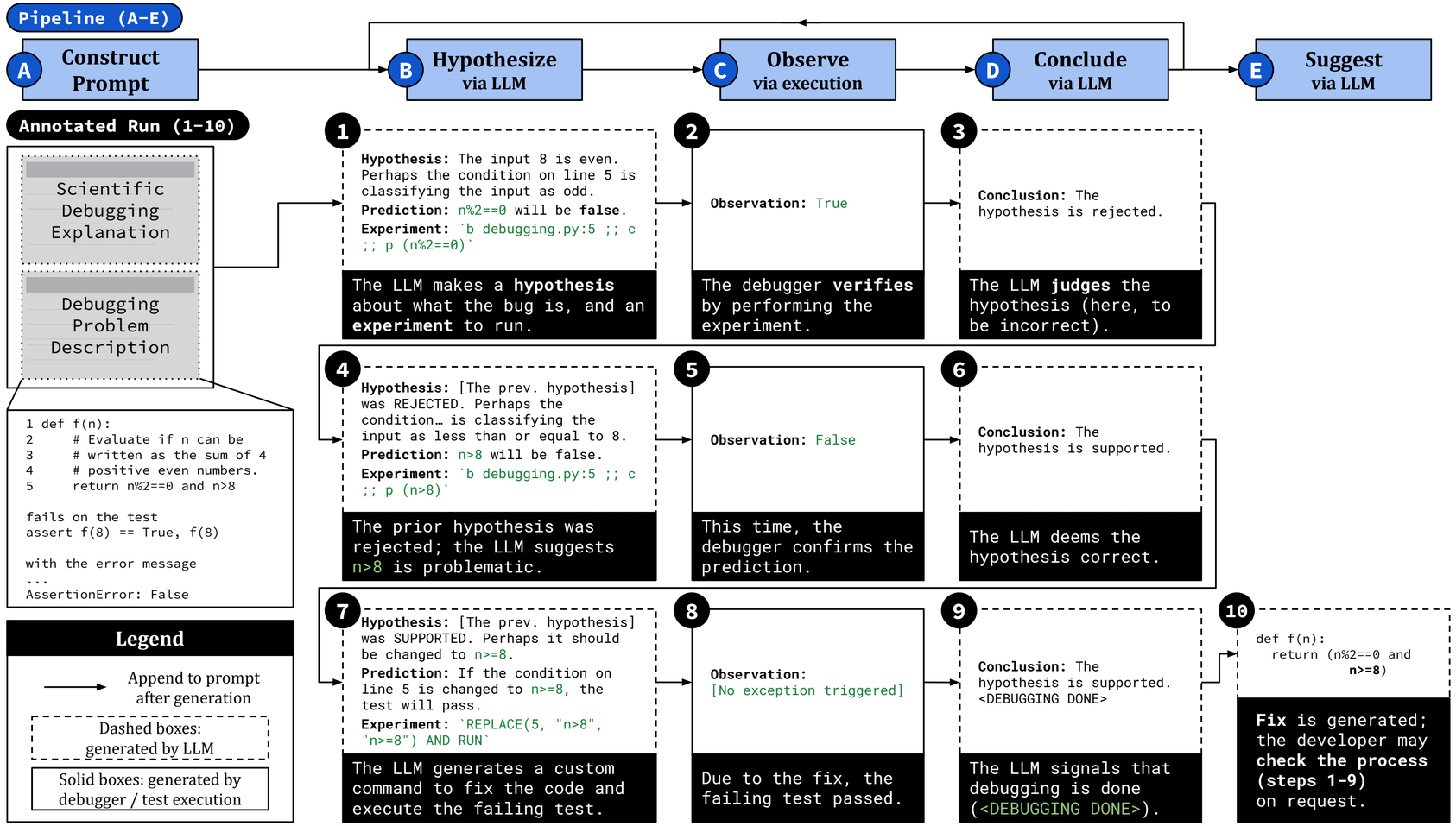
AutoSD: Zero-shot Automated Debugging

(back in 2022-2023, without knowing)

- AutoSD: an APR technique that performs Scientific Debugging
  - Zero-shot: only instructions about SD are given
  - Tool usage: it has access to debuggers to evaluate its hypotheses
- Scientific Debugging (Zeller, 2009)
  - Debugging should systematically follow the process of scientific discovery: hypothesis, design of experiments, observation, and conclude or refine.
- Our primary aim was to make APR **explainable**. We thought the SD process can be an explanation in itself.

AutoSD: Zero-shot Automated Debugging

Kang et al., EMSE 2025 (to appear)



Agentic Design Elements in AutoSD

- Use of debuggers: AutoSD uses `pdb` to repair Python programs (mutated HumanEval). Since Python is interpreted, its debugger is relatively straightforward to use. Similarly, Java is friendly to AutoSD.
  - We later tried to port AutoSD to C/C++; `gdb` was much pickier.
- DSL for Code Edit: a fairly simple set of edit commands, based on line numbers and substring matching
  - Multi-hunk, multi-file patches may require a more sophisticated approach.
- Test Execution: the efficacy of the generated patch is validated using test execution.
  - Again, we benefit from Python's simple test framework.

Second Agent had Function Call Support

AutoFL by Kang & An, FSE 2024

- We also wanted to apply LLMs to FL (hammer and nail problem) but did not know how back in 2022~2023.
  - Mainly because of the context length limit. You cannot ask LLM to pinpoint a code location that has not been included in the context!
  - Wu et al. (https://arxiv.org/abs/2308.15276) assumes we have the faulty method, and tried statement level localization within the method (because a method fits the context length).
  - We tried sampling top-k methods from SBFL then querying the LLM - didn't perform particularly well.
  - How do we go **repository-level**?

AutoFL: LLM based FL

Kang, An & Yoo, FSE 2024 (https://arxiv.org/abs/2308.05487)

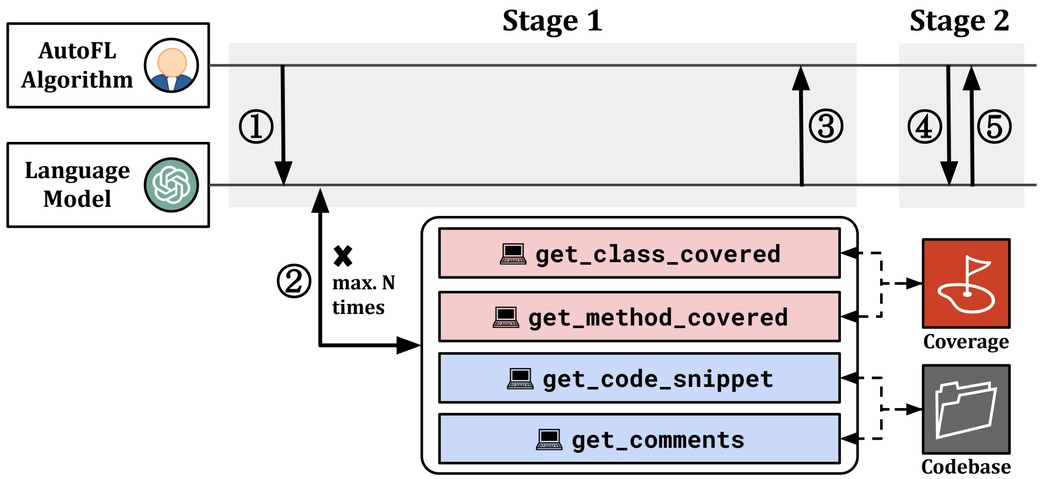


Figure 1: Diagram of AutoFL. Each arrow represents a prompt / response between components, with the circled numbers indicating the order of interactions. Function invocations are made at most N times, where N is a predetermined parameter of AutoFL.

1. Prompt LLM to start by looking at the list of classes covered by the failing test.

2. Up to 10 function calls.

3. After 10 calls, or when it is ready, the LLM generates a user-facing explanation.

4. We prompt the LLM to come up with buggy methods.

5. The model responds.

AutoFL: LLM based FL

Kang, An & Yoo, FSE 2024 (https://arxiv.org/abs/2308.05487)

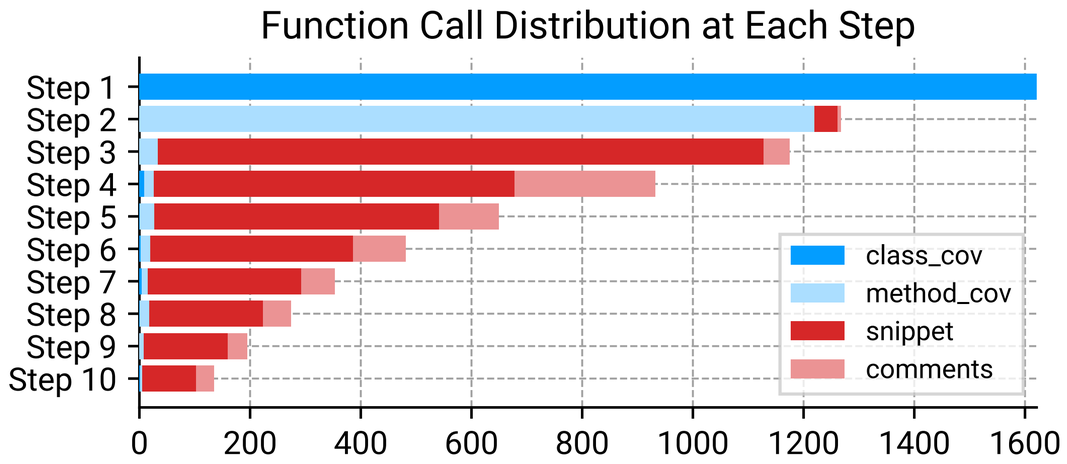


Figure 5: Function call frequency by step over all five runs of AutoFL. The total length at each step decreases as AutoFL can stop calling functions at any step; e.g. about 400 AutoFL processes stopped calling functions after the first step.

- Patterns do seem reasonable.
- The model gradually narrows down the scope, eventually looking at snippets (i.e. methods).
- Occasionally it looks at comments.

Agentic Design Elements in AutoFL

- Use of function calls: compared to AutoSD, there are more choices for AutoFL in terms of which function to call.
  - The functions do require some pre-processing, but these are engineering cost that is not beyond usual testing scenarios in CI environments.
- Decision to terminate: AutoFL decides when Stage 1 can be terminated.
  - This is more difficult than the case with AutoSD, which operates within the framework of Scientific Debugging. In general, knowing when it is done is difficult, when given an open task.

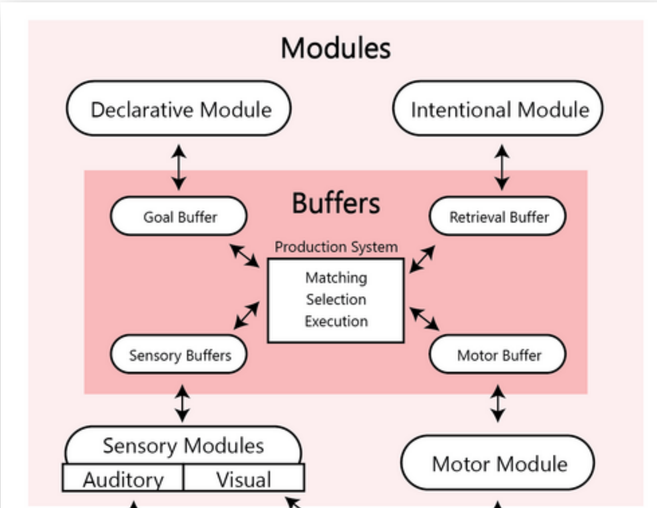
Designing Agentic Systems

- With AutoSD and AutoFL, the agent architecture as well as the design purpose is straightforward.
  - Both have a well defined operational framework (scientific debugging, or repository navigation)
  - It is relatively clear which tools should be given.
- How should we break down a complicated task so that it can be handled by agents?
  - This is a wide open topic, and I suspect that traditional software architecture can help but not have all the answers.
  - We will consider two candidates: cognitive architecture, and domain models

Cognitive Architecture

(“How can we model human cognition computationally?”)

- Theory and computational instantiation of how human mind is structured; can be a blueprint for an intelligent agent (wikipedia) - this is a loosely defined term, rather than a rigid theory.
- There are many widely known examples
  - ACT-R (Adaptive Control of Thoughts-Rational): developed at CMU
  - SOAR (State-Operator-and-Result): led by Alan Newell



DroidAgent: Planner-Actor-Reflector

Yoon et al., ICST 2024 (https://arxiv.org/abs/2311.08649)

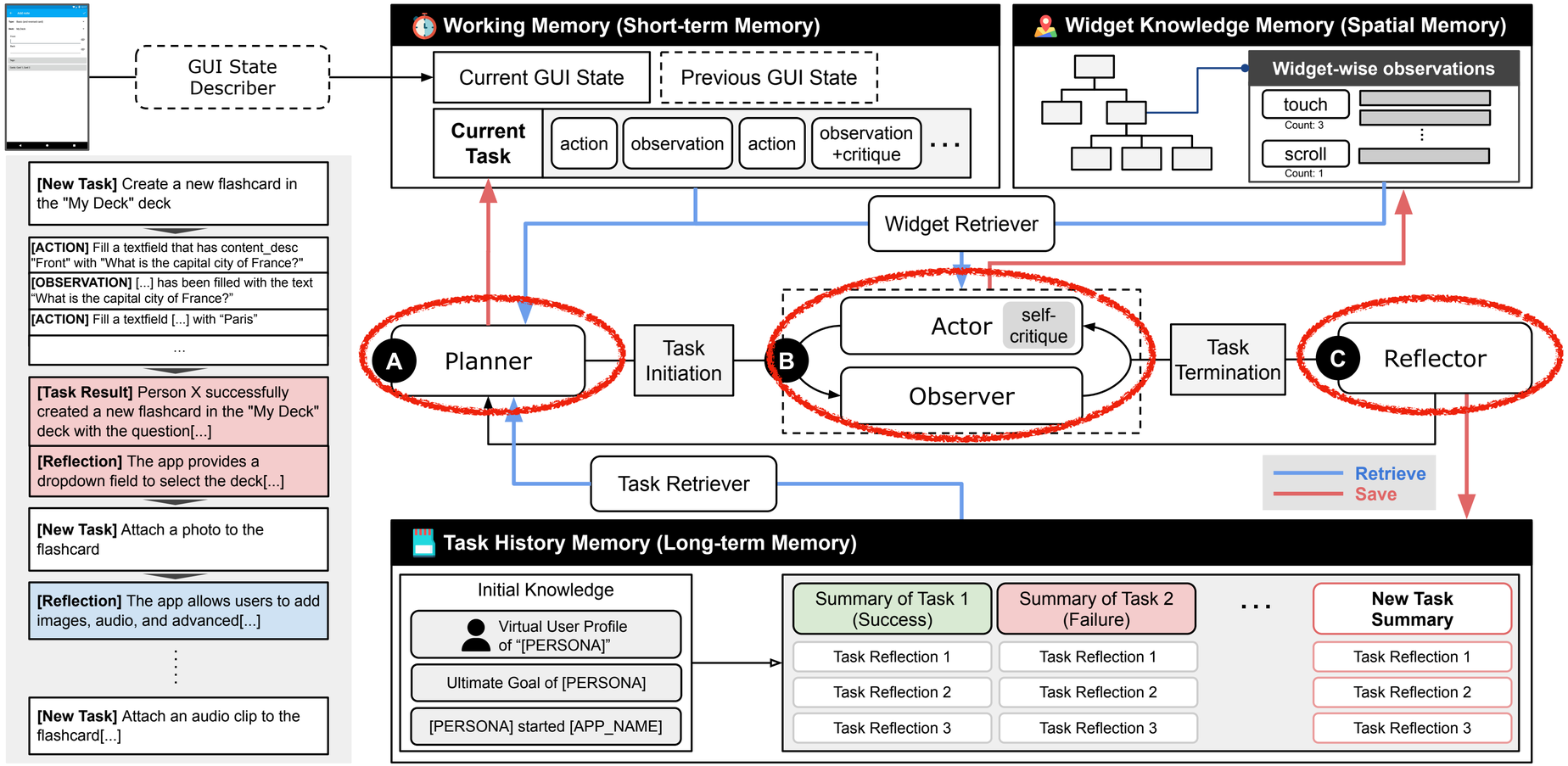


Fig. 1. Overview of DROIDAGENT with a task example.

DroidAgent: Planner-Actor-Reflector

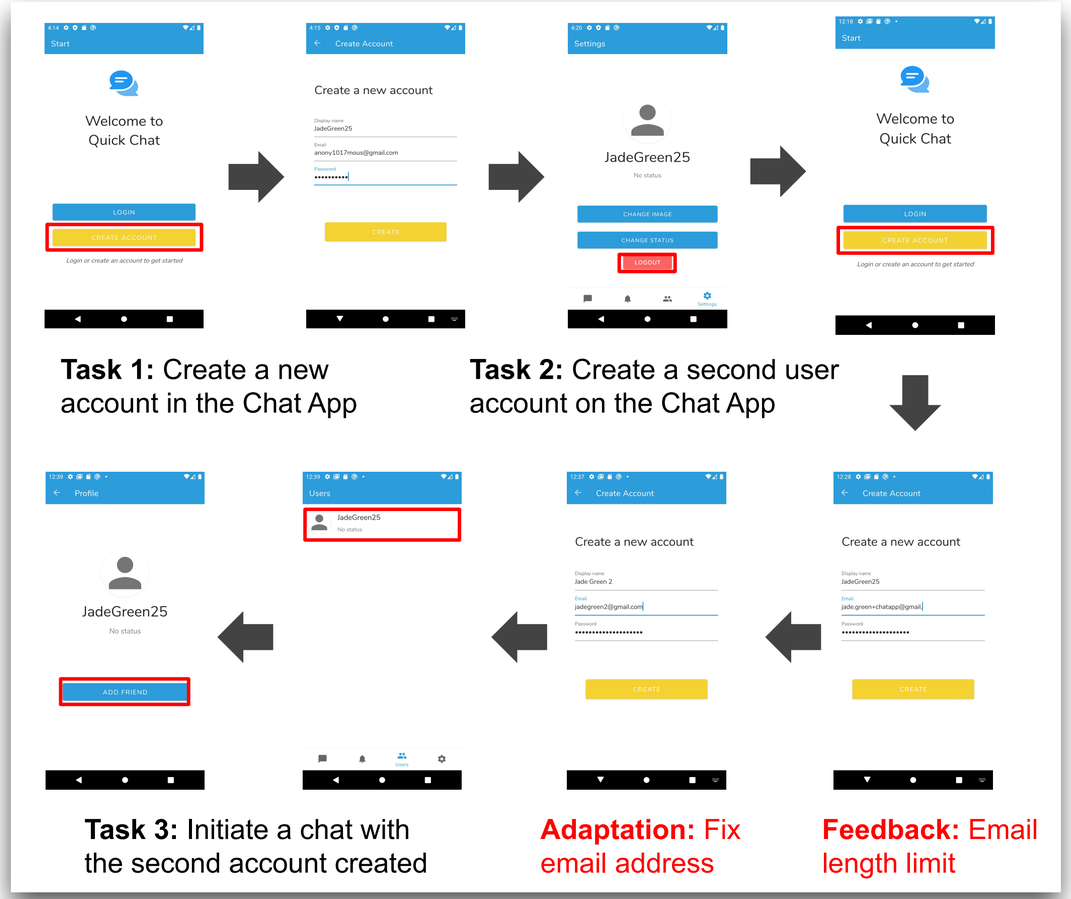
Yoon et al., ICST 2024 (https://arxiv.org/abs/2311.08649)

```
Reasoning about Jade Green's new task: To provide a diverse and
realistic task that makes use of the core functionality of the app, Jade
 Green should try to add an audio clip to a flashcard, which is an
important feature of AnkiDroid to enhance learning efficiency. This task
 is not too difficult as it is similar to the previous task of adding an
 image to a flashcard.

Jade Green's next task: Add an audio clip to a flashcard.
```

Planner worked well because LLMs could anticipate what app functionalities are without much problem.
(Quiz: how would you improve the planner?)

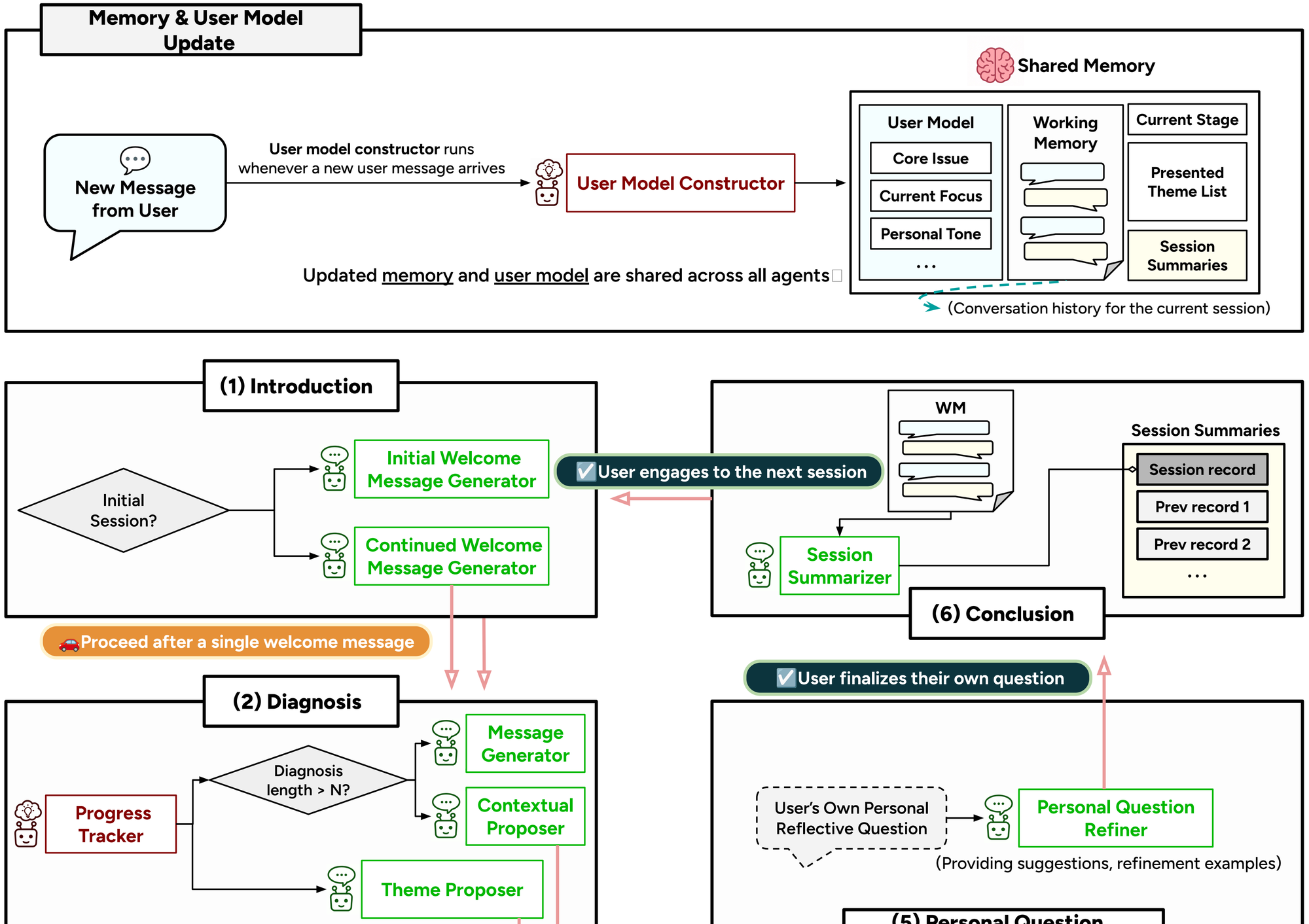
Actor/Observer worked well because GUI is a stateful system.

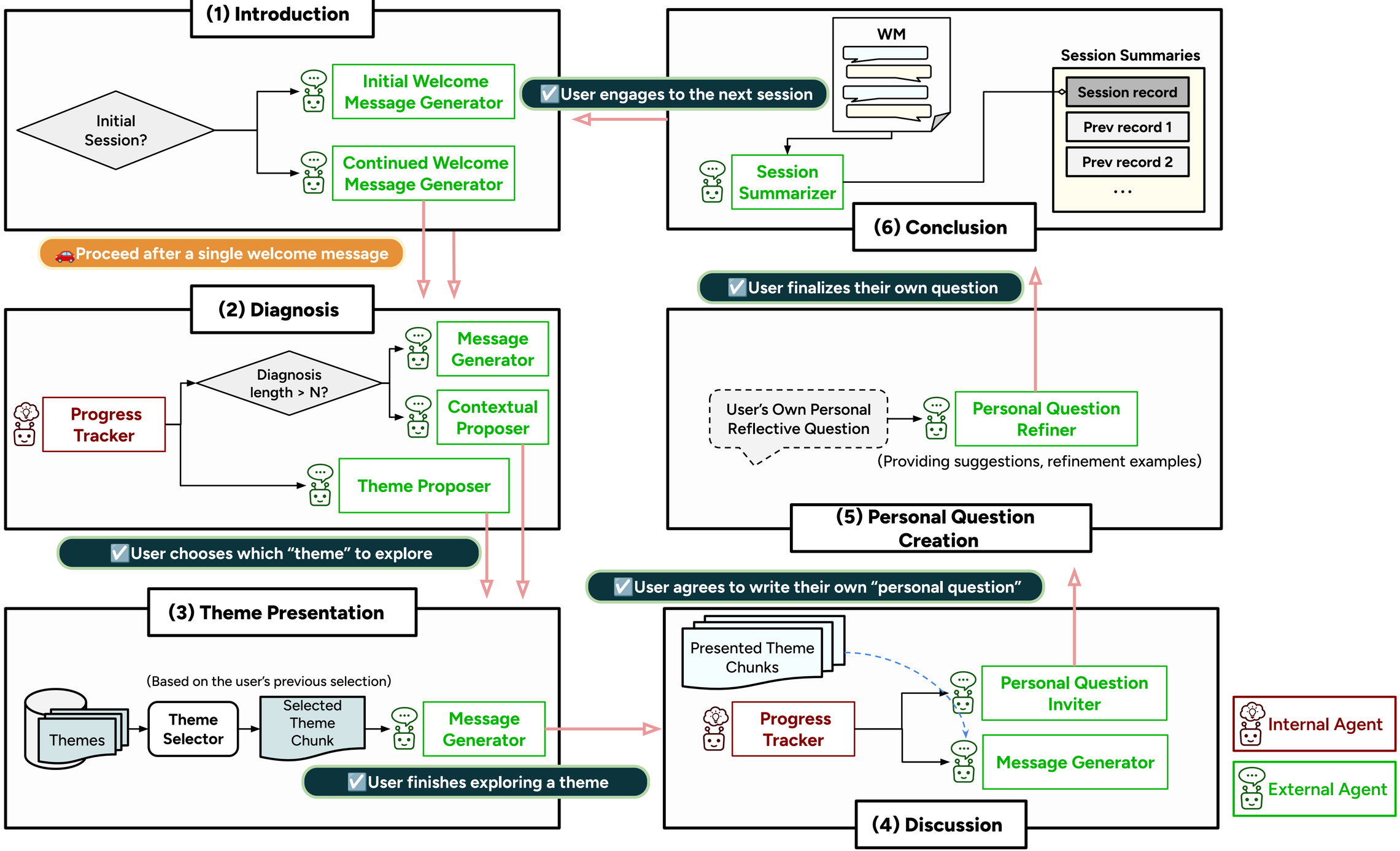


Domain Models

(Work in Progress)

- Sometimes, the exact agent workflow is given by the domain.
- **SIA** (Structured Interview Agent) is an ongoing project at COINSE@KAIST; our aim is to build multi-agent system that can perform structured conversations/ interviews.
    - For example, there are frameworks like Motivational Interview (MI) or Cognitive Behavioral Therapy (CBT) in psychology. You are expected to go through certain stages of conversation, each with different focus or aim.


Memory & User Model Update
Shared Memory
New Message from User
User model constructor runs whenever a new user message arrives
User Model Constructor
User Model
Core Issue
Current Focus
Personal Tone
...
Working Memory
Current Stage
Presented Theme List
Session Summaries
Updated memory and user model are shared across all agents
(Conversation history for the current session)
(1) Introduction
Initial Session?
Initial Welcome Message Generator
Continued Welcome Message Generator
Proceed after a single welcome message
(2) Diagnosis
Progress Tracker
Diagnosis length > N?
Message Generator
Contextual Proposer
Theme Proposer
User engages to the next session
WM
Session Summarizer
Session Summaries
Session record
Prev record 1
Prev record 2
...
(6) Conclusion
User finalizes their own question
User's Own Personal Reflective Question
Personal Question Refiner
(Providing suggestions, refinement examples)
(5) Personal Question


(1) Introduction
Initial Session?
Initial Welcome Message Generator
Continued Welcome Message Generator
🚗Proceed after a single welcome message
(2) Diagnosis
Progress Tracker
Diagnosis length > N?
Message Generator
Contextual Proposer
Theme Proposer
☑️User chooses which "theme" to explore
(3) Theme Presentation
(Based on the user's previous selection)
Themes
Theme Selector
Selected Theme Chunk
Message Generator
☑️User finishes exploring a theme
☑️User agrees to write their own "personal question"
Presented Theme Chunks
Progress Tracker
Personal Question Inviter
Message Generator
(4) Discussion
☑️User finalizes their own question
User's Own Personal Reflective Question
Personal Question Refiner
(Providing suggestions, refinement examples)
(5) Personal Question Creation
☑️User engages to the next session
WM
Session Summarizer
Session Summaries
Session record
Prev record 1
Prev record 2
...
(6) Conclusion
Internal Agent
External Agent

Finally, about uncertainty & ood.

Why do we care about uncertainty?

- ML systems are the most representative “non-testable programs” [1].
  - In that we use ML mainly because we do not know the answer ourselves (i.e., there is no easy and known computation that results in the answer).
- Without oracles in hand, testing becomes slightly… weird?:
  - Metamorphic Testing: essentially we go after relaxed oracles
  - Prioritization: we accept the high cost of oracles, but we only look at results that are likely to be incorrect

[1] Weyuker, E. J. On Testing Non-Testable Programs. The Computer Journal 25, 4 (November 1982), 465–470,

Uncertainty in Machine Learning 🌀

Two major sources

- Aleatoric Uncertainty (우연적 불확실성): randomness that cannot be avoided even under perfect knowledge
  - Results of coin flips, physical noise in sensor circuits, etc…
- Epistemic Uncertainty (인식론적 불확실성): uncertainty that **can** be reduced if you **learn harder** (typically by adding more training data, or changing the model)
  - Low accuracy due to too small a model or insufficient training data

Quantifying Uncertainty

- Exact, analytical quantification is only possible with mathematically grounded models that allows you to compute the confidence intervals.
- Training-based Approaches: MCMC, Bayesian Neural Networks, etc
  - Intuitively, try to capture and learn the variance in data during training
- Training-free Approaches: Bayesian Dropout, Gradient-based, Test-time Data Augmentation, etc.
  - Intuitively, try to inject small perturbation during inference and observe the variance (i.e., measure how “brittle” the model decision is “around” the original result)

A Tangential Approach
(that has been successful… so far)

- Out-Of-Distribution-ness (OOD) is correlated with uncertainty:
  - If a new sample is close to the mean of training data distribution, the model will have low uncertainty (provided that the training has been effective)
  - If a new sample comes from an unseen distribution, the model is likely to show high epistemic uncertainty!
- This was the core idea behind Surprise Adequacy [2]

[2] Kim, J., Feldt, R., and Yoo, S. Guiding deep learning system testing using surprise adequacy. ICSE 2019, pp. 1039–1049.

For large pre-trained LLMs, such as GPT4o, what is the training data?

(answer: the entire ~~human knowledge~~ internet)

(or, even worse, some task-specific accuracy based on **emergent** inference capabilities…)

For large pre-trained LLMs, such as GPT4o, what is the model accuracy?

(answer: autoregressive loss against… the entire ~~human knowledge~~ internet)

LLM Input Distribution: No Global Frame of Reference

- Too big to handle, both conceptually and practically, i.e., too big to use as a reference to measure the OOD of a new sample.
- It would be nice to have a more manageable, but **sufficiently representative**, frame of reference.
  - Fine tuning dataset (if you have used fine-tuning)
  - Task-specific reference set (but how representative is it?)

Test Oracle

- For traditional programs, oracles are predicates that, given a stimulus to the system under test, tell you whether the observed behavior are as expected. They are discrete, Boolean.
- For ML systems, oracles are probability distributions that, given a new sample, tell you how likely it is that the resulting behavior are as expected. They are **continuous and approximate** [3].

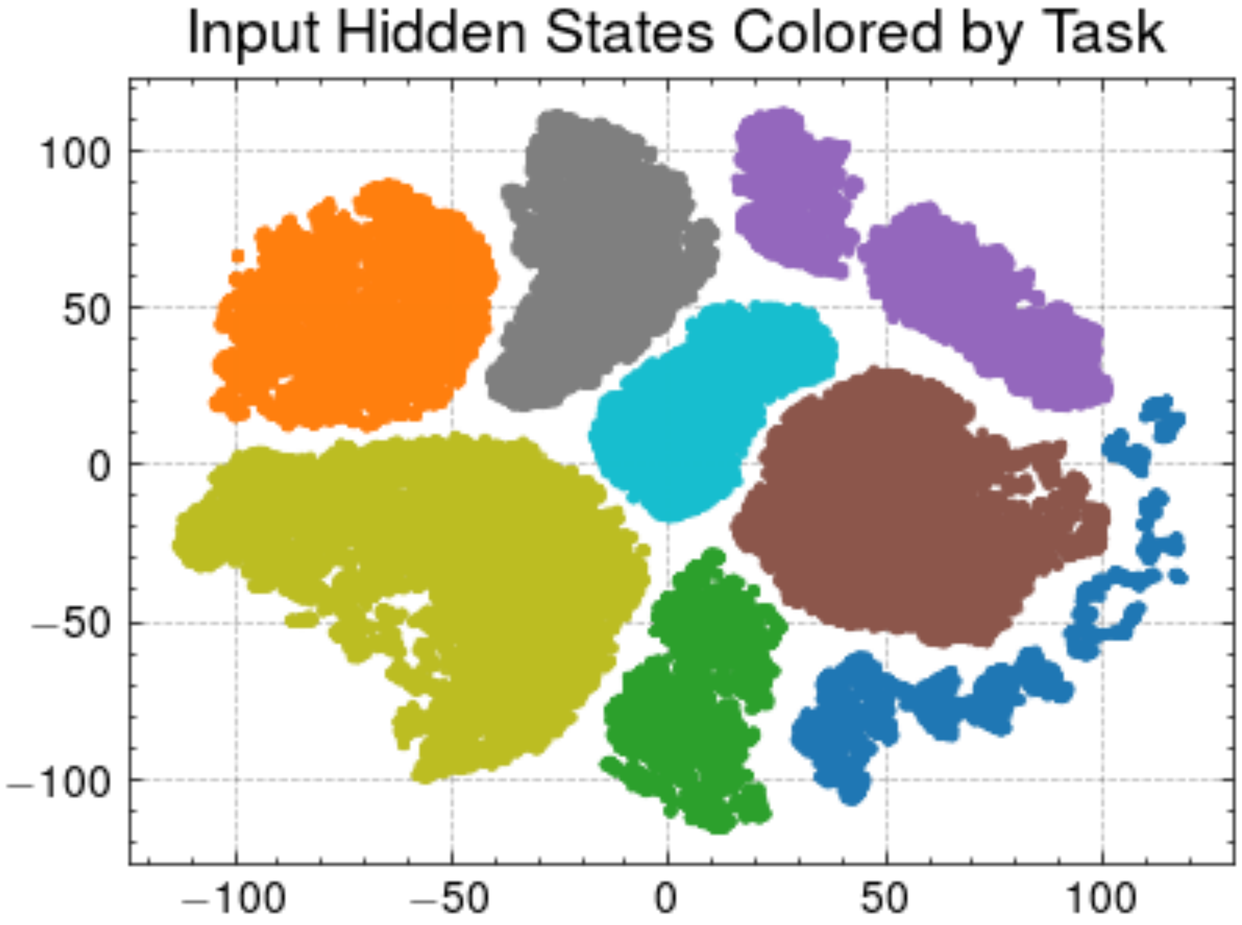
Definition 2.5 (Probabilistic Test Oracle). *A* probabilistic test oracle $\tilde{D} : T_A \mapsto [0, 1]$ *maps a test activity sequence into the interval* $[0, 1] \in \mathbb{R}$.

[3] Barr, E., Harman, M., McMinn, P., Shahbaz, M., and Yoo, S. The oracle problem in software testing: A survey. IEEE Transactions on Software Engineering 41, 5 (May 2015), 507–525.

Clotho: Task-Specific Test Adequacy

https://arxiv.org/abs/2509.17314

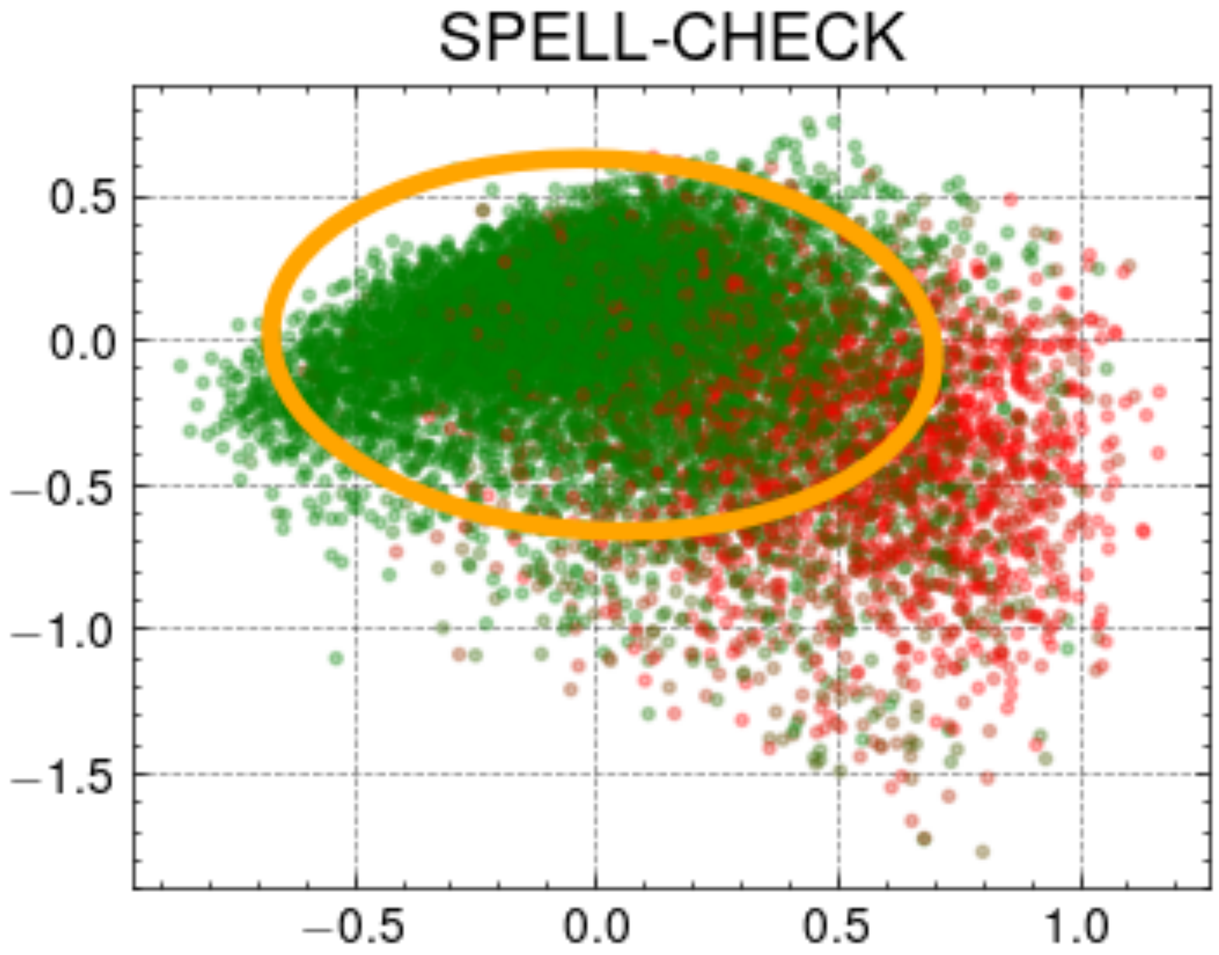
- Latent space at the last token of input (i.e., before generating any answer) shows that tasks form different groups.



Clotho: Task-Specific Test Adequacy

https://arxiv.org/abs/2509.17314

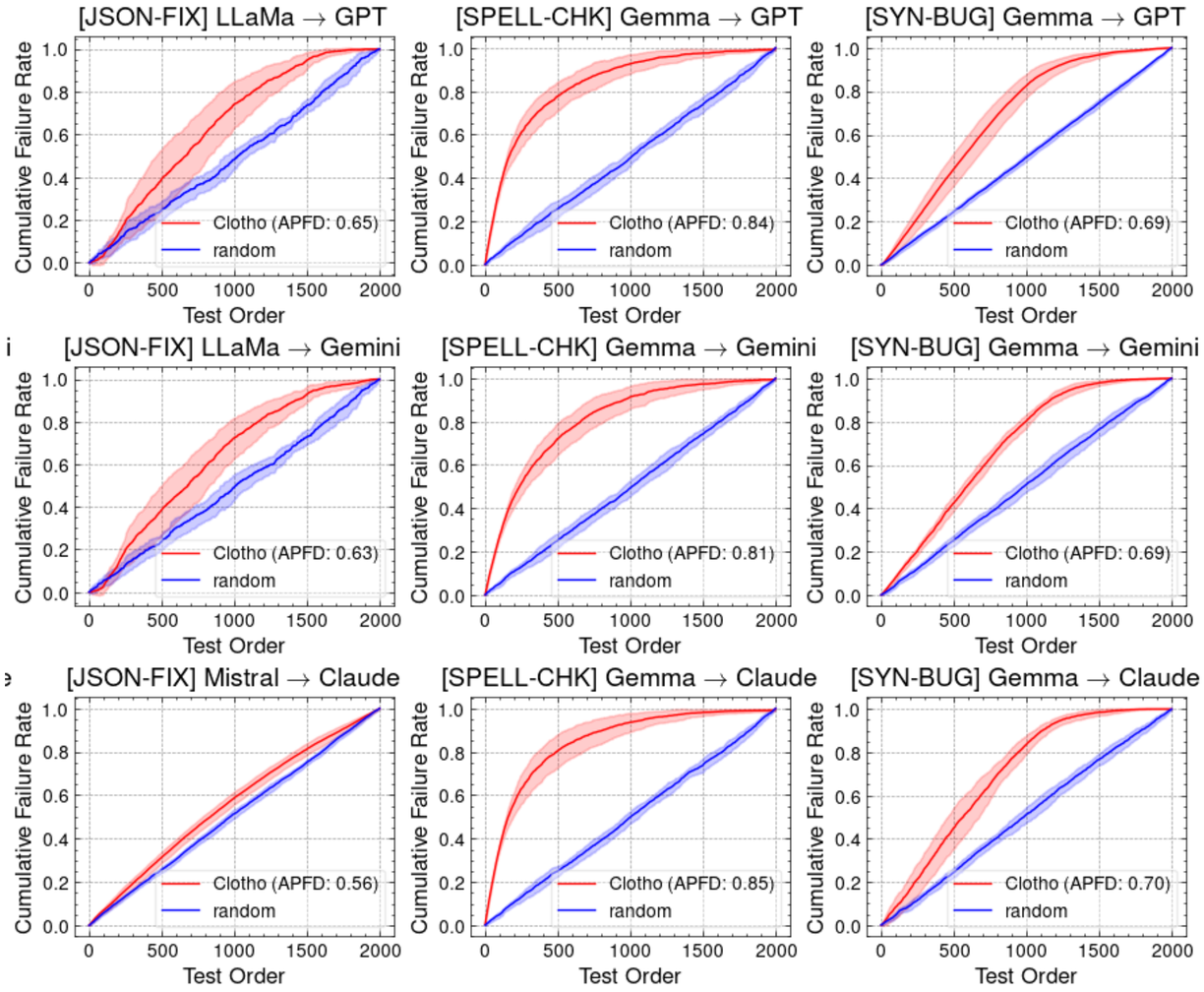
- Within each task cluster, inputs that eventually result in passing answers and failing answers form different distributions.
- OOD from passing distribution == Surprise Adequacy!



Clotho: Task-Specific Test Adequacy

https://arxiv.org/abs/2509.17314

- We can prioritize inputs based on how out-of-distribution they are from the passing distribution - early failure!
- Surprisingly, the adequacy transfers to larger model (measure with local, small open model —> prioritize —> run with cloud, large closed model)



A Thought Experiment

John Searle, “Mind, Brains, and Programs” in 1980

- Suppose we have a computer program that behaves as if it understands Chinese language.
- You are in a closed room with the AI program source code.
- Someone passes a paper with Chinese characters written on it, into the room.
- You use the source code as instruction to generate the response to the input, and sends the response out of the room.
- Do you understand Chinese language, or not?

Stochastic Parrot?

- Among other risks, authors ask whether LLMs actually “understand” anything.
- What do you think?
- The internal design is clearly a statistical language model, i.e., it says what is the most likely, not what is the correct.

Check for updates

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on the web, carrying out pre-development exercises evaluating how the planned approach fits into research and development goals and supports stakeholder values, and encouraging research directions beyond ever larger language models.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**.

ACM Reference Format:
Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜. In *Conference on Fairness, Accountability, and Transparency (FAccT '21), March 3–10, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3442188.3445922

1 INTRODUCTION

One of the biggest trends in natural language processing (NLP) has been the increasing size of language models (LMs) as measured by the number of parameters and size of training data. Since 2018 alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learning systems [129], we encourage the research community to prioritize these impacts. One way this can be done is by reporting costs and evaluating works based on the amount of resources they consume [57]. As we outline in §3, increasing the environmental and financial costs of these models doubly punishes marginalized communities that are least likely to benefit from the progress achieved by large LMs and most likely to be harmed by negative environmental consequences of its resource consumption. At the scale we are discussing (outlined in §2), the first consideration should be the environmental cost.

Just as environmental impact scales with model size, so does the difficulty of understanding what is in the training data. In §4, we discuss how large datasets based on texts from the Internet overrepresent hegemonic viewpoints and encode biases potentially damaging to marginalized populations. In collecting ever larger datasets we risk incurring documentation debt. We recommend mitigating these risks by budgeting for curation and documentation at the start of a project and only creating datasets as large as can be sufficiently documented.

As argued by Bender and Koller [14], it is important to understand the limitations of LMs and put their success in context. This not only helps reduce hype which can mislead the public and researchers themselves regarding the capabilities of these LMs, but might encourage new research directions that do not necessarily depend on having larger LMs. As we discuss in §5, LMs are not performing natural language understanding (NLU), and only have success in tasks that can be approached by manipulating linguistic form [14]. Focusing on state-of-the-art results on leaderboards without encouraging deeper understanding of the mechanism by which they are achieved can cause misleading results as shown

*Joint first authors

FAccT '21, March 3–10, 2021, Virtual Event, Canada
ACM ISBN 978-1-4503-8309-7/21/03.

Managing Expectations

- Should we treat LLMs as software components with real intelligence?
- Or a black-box that can generate surprisingly meaningful token sequences?
- (Multi-) Agents = shorter contexts for more focused decision making, orchestrated by human-designed workflow.
  - Fits both world-view, while providing us safeguards in the near future.

Summary

- LLMs are statistical language models with significant emergent capabilities.
- There are many In-Context Learning techniques that can be adapted for software engineering tasks.
- However, there are also major open issues.
  - Design: What are the design principles that should guide development of agentic systems?
  - Testability: What are the oracles for LLMs? How do we generate inputs?