

**Kirill Gelvan\*, Igor Slinko, Felix Steinbauer, Egor Bogomolov, Yaroslav Zharov**

## Why and how?

- ▶ LLMs have **limited context length**
- ▶ SWE Agents often call for tools with **unnecessarily long outputs**
- ▶ The **latent space** of embeddings is **much denser** than the discrete space of tokens  
⇒
- ▶ Let's learn to compress tool outputs and use only the required information for the subsequent steps

The diagram illustrates the proposed tool-use framework for LLM agents, comparing a baseline approach with the proposed method.

**Baseline Approach (Base agent's step 2):**

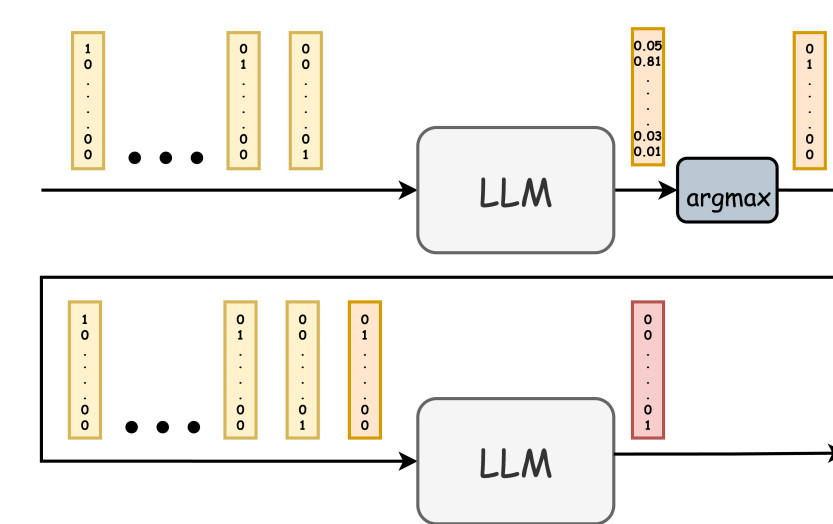
- The agent receives a prompt: "fix my error in ./dir".
- The agent makes a tool call: "ls -la ./dir".
- The Environment returns an observation:  $\langle 5000 \text{ tokens} \rangle$ .
- The LLM processes the observation, but it fails due to an "Error: Out of context length!" because the observation is too long (5000 tokens).

**Proposed Framework (Our agent's step 2):**

- The agent receives the same prompt: "fix my error in ./dir".
- The agent makes the same tool call: "ls -la ./dir".
- The Environment returns the same observation:  $\langle 5000 \text{ tokens} \rangle$ .
- The observation is first processed by a **Compressor**, which outputs **Compressed observation (256 embeddings)**.
- The LLM processes the compressed observation, successfully outputting "Next tool call..." (indicated by a green checkmark).

The diagram highlights that the proposed framework successfully handles long observations by using a compressor to reduce the token count, avoiding the "Out of context length" error.

## Hard Tokens (usual)



SQuAD[2], context embed	F1
Baseline — hard tokens	0.71

**Table:** Hard tokens technique

The diagram illustrates the process of generating a single output token from a sequence of input tokens using an LLM. It is divided into two horizontal sections.

**Top Section:** Shows a sequence of input tokens (1, 0, ., ., ., 0, 0) and a sequence of output tokens (0.01, 0.03, 0.01, 0.04, 0.03, 0.01). The input tokens are grouped into a box labeled "LLM", which produces the output tokens.

**Bottom Section:** Shows the same input sequence, but the output tokens are averaged (AVG) to produce a single output token (0.02). The input tokens are grouped into a box labeled "LLM", which produces the output tokens. The output tokens are then averaged (AVG) to produce a single output token (0.02).

SQuAD[2], context embed	F1
Soft-embedded online	0.17
Soft-embedded online, avg $\times 2$	0.11
Soft-embedded "regenerate-llm", avg $\times 2$	0.16

**Table:** Soft-embedded techniques

- ▶ No, we cannot!
- ▶ The scores drop dramatically (>50%) just by using continuous representations (without avg)
- ▶ In order to get to the latent space, **we need training**

The diagram illustrates the pre-training stage for Qwen3, divided into two main components: Qwen3 as Compressor and Qwen3 as Decoder.

**Qwen3 as Compressor:** This component takes **input tokens** (represented by a sequence of white boxes) and processes them through an **Embedding Layer** (blue box) and a **LoRA** layer (orange box). The output is a sequence of **memory tokens** (represented by a sequence of pink boxes). A bracket labeled **memory slots** indicates a subset of these memory tokens, with the last token labeled **ST** (Start Token).

**Qwen3 as Decoder:** This component takes the **memory slots** (pink boxes) and the **ST** token as input. It processes them through an **Embedding Layer** (blue box) and a **LoRA** layer (orange box). The output is a sequence of **answer tokens** (represented by a sequence of white boxes). A bracket labeled **teacher-forcing** indicates the relationship between the **ST** token and the first answer token.

**Pre-training stage:** The tasks are distinguished by a **special** token (represented by a blue box with a robot icon). The tasks are:

- 50% of data is **AutoEncoding (AE)**: answer tokens are input tokens
- 50% of data is **Language Modeling (LM)**: answer tokens are continuation tokens

### Fine-Tuning stage

- ▶ 100% of the data is from your own task. You may add a prompt after memory tokens as well

Note that only the "compressor" is trained, while **generation** is done by an **untouched Qwen** model!

## Can we decompress texts after pre-training?

Dataset	Model	BLEU
PWC [3]	Mistral-7B	99.1
	Llama-2-7B	99.5
SQuAD (ours)	Qwen3-8B	98.1

**Table:** Text decompression on PWC and SQuAD

## Can we solve Question-Answering task after fine-tuning?

Model	Compression	Exact Match (%)	F1
Mistral-7B	$\times 1$	49	68
LoRA-FT Mistral-7B	$\times 1$	59	65
ICAE-FT Mistral-7B	$\times 1.7 \pm 0.7$	<b>69</b>	<b>73</b>

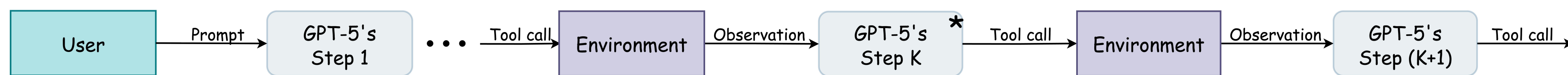
**Table:** ICAE averaging on SQuAD

### How can we interpret this?

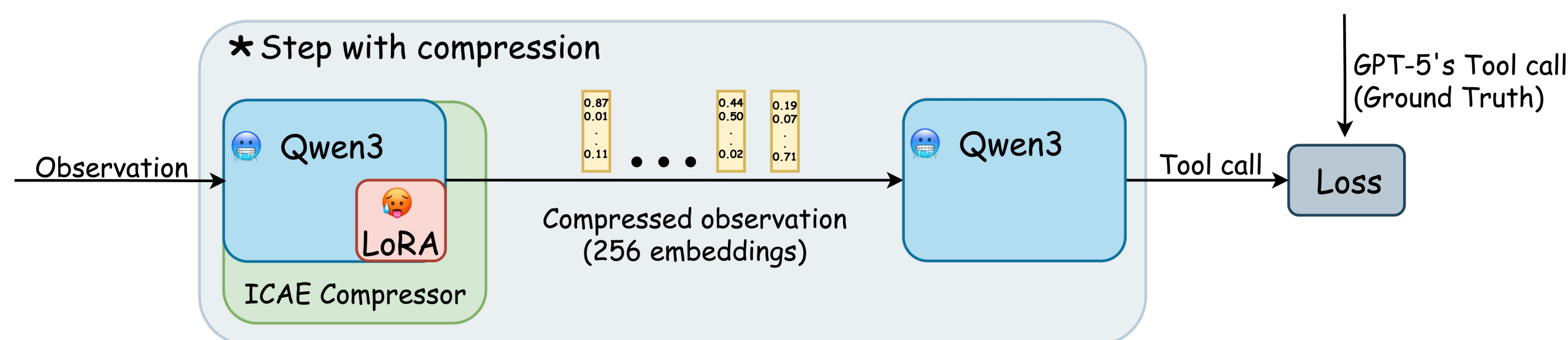
- ▶ ICAE is able to decompress texts almost perfectly
- ▶ ICAE works with newer models and different datasets
- ▶ But does it work in an agentic setup?

## 1 – Pre-train ICAE on general text corpora

## 2 – Get trajectories from a strong model (GPT-5)



### 3 — Fine-Tune Qwen3's LoRA on steps with compression:



## Results

- **Latency:** ICAE compression shows a **10% faster** mean tool-call generation time than vanilla Qwen3-8B
- **Token-wise accuracy:** Qwen3-8B with and without compression perform **on par**
- **Resolved on SWE-bench Verified:** The model with compression **resolves fewer than 50%** of the original number of issues.

## Hypotheses

- ▶ **Representation–behavior mismatch:** The ICAE encoder boosts token-level accuracy slightly, but perturbs decoder behavior for tool use, causing fewer end-to-end “resolved” completions
- ▶ **Compression trade-off:** Faster inference trims useful context or exploration, improving latency but reducing robustness on multi-step tasks required to count as “resolved”
- ▶ **Training dynamics / overfitting:** The low resolved count for the higher-accuracy variant suggests overfitting to labels rather than to execution reliability

[1] Neil Chowdhury et al. Introducing SWE-bench Verified. 2024.

[2] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text".

[3] Tao Ge et al. "In-context Autoencoder for Context Compression in a Large Language Model".

