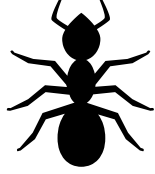




Small , open-source 
and multilingual 
function-calling agents

Why function-calling?

- Natural language user interfaces
 - Voice assistants like Alexa, Siri, etc.
 - Chatbots in websites
 - Video game NPCs
 - Accessibility

Octopus v2 Paper

<https://arxiv.org/pdf/2404.01744>

Octopus v2: On-device language model for super agent

Wei Chen[†] *
Stanford University
{weichen6}@stanford.edu

Zhiyuan Li[†]
Stanford University
{zhiyuan8}@stanford.edu

Abstract

Language models have shown effectiveness in a variety of software applications, particularly in tasks related to automatic workflow. These models possess the crucial ability to call functions, which is essential in creating AI agents. Despite the high performance of large-scale language models in cloud environments, they are often associated with concerns over privacy and cost. Current on-device models for function calling face issues with latency and accuracy. Our research presents a new method that empowers an on-device model with 2 billion parameters to surpass the performance of GPT-4 in both accuracy and latency, and decrease the context length by 95%. When compared to Llama-7B with a RAG-based function calling mechanism, our method enhances latency by 35-fold. This method reduces the latency to levels deemed suitable for deployment across a variety of edge devices in production environments, aligning with the performance requisites for real-world

Octopus v2 Prompt Template

Below is the query from the users, please choose the correct function and generate the parameters to call the function.

Query: {query}

Response: <oc_i>(param1, param2, ...)<oc_end>

Function description: {function_description}

Functional Tokens

<oc_i>:

Function classification and knowledge of function definitions

<oc_end>:

Early stopping criterion

```
new_tokens = ["<oc_1>", "<oc_2>", "<oc_3>", "<oc_end>"]  
new_tokens = set(new_tokens) - set(tokenizer.vocab.keys())
```

```
tokenizer.add_tokens(list(new_tokens))  
model.resize_token_embeddings(len(tokenizer))
```

Generating Training Data

- 3 functions to interact with a calendar application
 - Create calendar entry
 - Delete calendar entry
 - List calendar entries
- Generating commands from templates
 - Handwritten lists of values
 - Python functions to generate values such as dates, times, durations

Generating Training Data

```
command_templates = [  
    "Create a new calendar entry for {activity} on {date} at {time} for {duration}",  
    "Create a calendar entry for {activity} on {date} at {time} for {duration}",  
    "Schedule {activity} on {date} at {time} for {duration} in the calendar",  
    "Schedule {activity} on {date} at {time} for {duration}",  
    "Enter {activity} for the {date} at {time} with a duration of {duration}",  
    "Enter {activity} for the {date} at {time} with a duration of {duration} in the calendar",  
    "Add a calendar entry for {activity} on {date} at {time} for {duration}"  
]  
  
command_templates_timed = [  
    "Create a new calendar entry on {date} at {time} for {timed_activity}",  
    "Create a calendar entry on {date} at {time} for {timed_activity}",  
    "Add {timed_activity} to the calendar on {date} at {time}",  
    "Add {timed_activity} to the calendar on {date} at {time}",  
    "Add a calendar entry for {timed_activity} on {date} at {time}",  
    "Schedule {timed_activity} in the calendar on {date} at {time}",  
    "Schedule {timed_activity} on {date} at {time}",  
]
```

Models

Qwen2-0.5B and Qwen2-1.5B:

- Smallest models of a family of open-source LLMs trained on partially multilingual data

SauerkrautLM-1.5b:

- Continuously trained on German data from Qwen2-1.5B
- Bilingual in English and German

Phi-3-mini-4k-instruct:

- Worst performance also due to strange tokenization

Supervised Fine-Tuning (SFT)

- Initial fine-tuning step
- Full fine-tuning with hyperparameter configuration from the paper
- Fine-tuned model already capable of classifying the correct function and generating the early stopping token
- Shortcomings in parameter extraction -> preference optimization?
 - Fine-tuned model can be used to generate rejected examples

Preference Optimization

- DPO following SFT didn't work at all
 - Model forgot about the structure and started to repeat itself
- ORPO following SFT worked much better
 - Results closest to a valid function call
 - Lost some classification accuracy
- Standalone ORPO also works
 - Not better than SFT in experiments

Model Evaluation

- Generation of additional test commands including ground truth parameters
- Classification accuracy:
 - Classification accuracy can be a separate metric
- Parameter extraction correctness:
 - Evaluation via LLM judge, which decides whether a parameter is valid or not
- Structural correctness:
 - Regex patterns to validate output and parameter formats (dates, times, etc.)

Results

```
Erstelle einen neuen Kalendertermin für ein Weihnachtsessen bei meinen Eltern am 24.12.2024 um 18 Uhr für 3 Stunden
<oc_1>("Weihnachtsessen bei meinen Eltern", "2024-12-24", "18:00", 180)<oc_end>
=====
SauerkrautLM-1.5b SFT 1 Epoch:
<oc_1>("Weihnachtssuch", "2025-12-24", "18:00", 18)<oc_end>
-----
Qwen2-0.5b SFT 1 Epoch:
<oc_1>("Weihn ccap", "2025-12-24", "One十八")<oc_end>
-----
SauerkrautLM-1.5b SFT 3 Epochs:
<oc_1>("Weihnachtsessen Bei meinem Elernzeit", "2025-12-26", "18:00", -)<oc_end>
-----
Qwen2-0.5b ORPO 3 Epochs:
<oc_1>("Weihnachtessen bei meinem an", "2024-12-24", '18:05', 96)<oc_end>
-----
SauerkrautLM-1.5b SFT 1 Epoch + ORPO 1 Epoch:
<oc_1>("Websites", "2024-12-24", "18:00", 180)<oc_end>
```

Results

Neuen Kalendertermin für Projektpräsentation am 24.06.2024 um 18 Uhr für 90 Minuten anlegen

```
<oc_1>("Projektpräsentation", "2024-06-24", "18:00", 90)<oc_end>
```

=====

SauerkrautLM-1.5b SFT 1 Epoch:

```
<oc_1>("Projektpräsentations", "2025-06-24", "18:00", 90)<oc_end>
```

Qwen2-0.5b SFT 1 Epoch:

```
<oc_1>("Projektpräsentation", "2023-05 -24", '18':00", 90)<oc_end>
```

SauerkrautLM-1.5b SFT 3 Epochs:

```
<oc_1>("Projektpräsentation", "2025-06-34", "18:00", 70)<oc_end>
```

Qwen2-0.5b ORPO 3 Epochs:

```
<oc_1>("Projektvortrag", "2024-05-34", '18:00', 70)<oc_end>
```

SauerkrautLM-1.5b SFT 1 Epoch + ORPO 1 Epoch:

```
<oc_3>("Projektpräsentation", "YYYY-MM-DD")<oc_end>
```

Results

Lege einen sechsstündigen Termin für eine Silvesterparty am 31.12.2025 ab 20 Uhr an

<oc_1>("Silvesterparty", "2025-12-31", "20:00", 360)<oc_end>

=====

SauerkrautLM-1.5b SFT 1 Epoch:

<oc_1>("Seminarvortrag", "2024-12-31ab", "20:00", 6)<oc_end>

Qwen2-0.5b SFT 1 Epoch:

<oc_1>("lesung Mittelhochdeutsch", "2024-12-31", "-20"<oc_end>

SauerkrautLM-1.5b SFT 3 Epochs:

<oc_1>("SilsterParty", "2024-12-39)<oc_end>

Qwen2-0.5b ORPO 3 Epochs:

<oc_1>("silereintrag", "2025-18-34", 'ab2eb9a76c')<oc_end>

SauerkrautLM-1.5b SFT 1 Epoch + ORPO 1 Epoch:

<oc_3>("Seminarvortrag", "YYYY-MM-DD9e")<oc_end>

Challenges

- Creating an AI generated set of commands that also sounds like real human input
- HuggingFace fine-tuning libraries take some time to get used to
 - Some models, such as Qwen2, require pad tokens to be added
- Track different runs with different approaches
 - Storing results and checkpoints requires a lot of disk storage

Conclusion/Learnings

- Classification works quite well (at least for 3 functions)
- Generating realistic training data is difficult
- Alignment/preference optimization is not a trivial follow up step to SFT
- Model must work reasonably reliably for evaluation to be useful
 - Classification and structural correctness ability can be evaluated early on
 - Parameter generation requires the model to produce somewhat correct outputs to make sense

Next Steps

- Increase the reliability of the model outputs
- Evaluate influence of training on additional functions for parameter generation capability
- Create a more diverse (AI-generated) dataset
- Apply the model to a real use case (e.g. digital tourist information)