AccGPT

A Chatbot for CERN Internal Knowledge

Florian Rehm, Verena Kain, Juan Manuel Guijarro, Sofia Vallecorsa

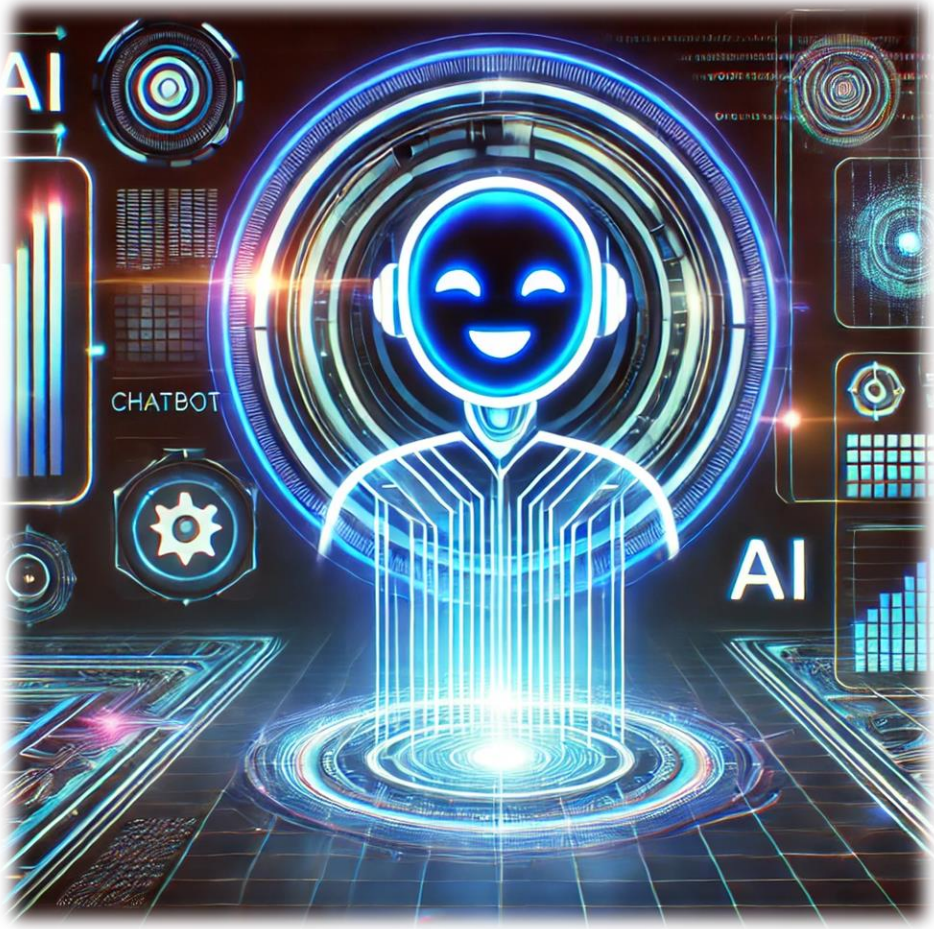
28.06.2024

Introduction and Definitions

- **AccGPT:** Accelerating Science – Generative Pretrained Transformer.
- **NLP** (Natural Language Processing): Enabling computers to understand and interact with human language.
- **Transformers:** Deep learning architecture with attention mechanisms; a key innovation in NLP.
- **GPT (Generative Pretrained Transformer):** State-of-the-art models using transformers to generate coherent text; applied in various (text-based) tasks.
- Inspired by ChatGPT and other chat-bot models (GPT4, Google Bard, …).

Agenda

- Why AccGPT?
- A few technical details.
- Some AccGPT examples.
- Current status and what’s next.



Why AccGPT?

Why AccGPT?

First step: Enhancing knowledge retrieval.

- Challenge: **CERN has many and HUGE data bases:**
  - >> 50 knowledge (web) domains for documentation.
    - Challenging to find information without knowing its location.
  - CERN Confluence (Wiki): > 1M pages.
  - CERN Document Server (CDS): > 500k documents.
  - CERN home: > 10k webpages.
  - CERNbox and more domains …

By GPT4

→ **Objective:** Leverage AccGPT to improve knowledge finding, user support, streamline development processes, and enhance onboarding experiences.

Why AccGPT? The List Goes on …

Numerous software frameworks such as FESA, UCAP, GeOFF, NXCALS, …

- With steep learning curves to use them.

  → AI assistance for learning and utilizing them.

Coding assistance:

- Many (internal) GitLab repositories.
- Aid in code development, understanding and debugging.
- Many documentations contain code.
  - *„Generate me a code template of …“*

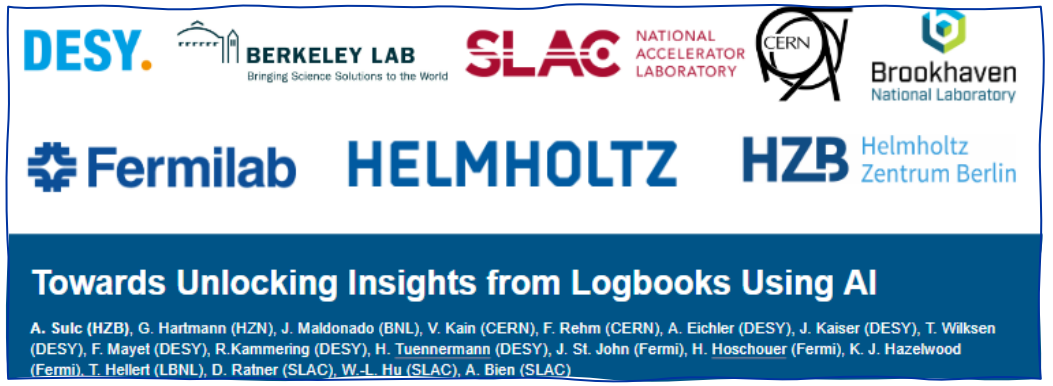
Future AccGPT applications:

- Enhancements to machine and shift logbooks.
- Further add-ons …

Future ++:

- AI assistant in the control room: conversational UI / controls.

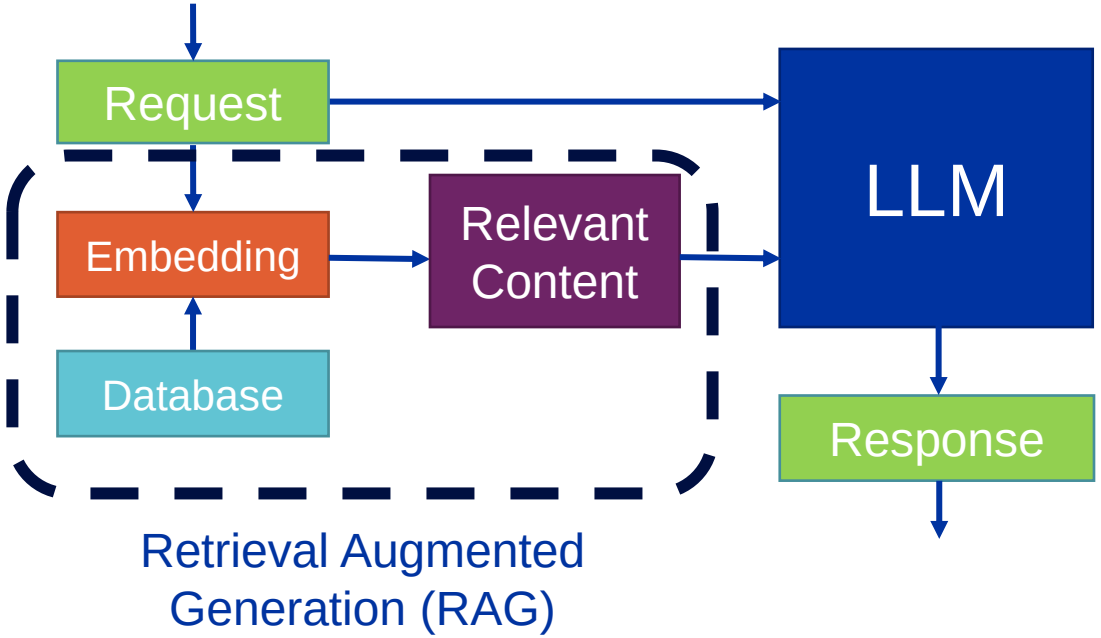
IPAC’24 - 15th International Particle Accelerator Conference



Inside of AccGPT

Inside of AccGPT – Conventional Setup

The (core) AccGPT pipeline*:



- **Based on two core models:**
  1. Embedding model:
     - Retrieves „relevant content“ from database.
  2. Large Language Model (LLM):
     - Formulates responses using the „relevant content“.
     - Focus on open-source models.
     - Possibility to use commercial models.
- **Accompanied by a self-created knowledge data base.**

* In reality the AccGPT pipeline has progressed and is more complex.

1. The Embedding Model

Function:

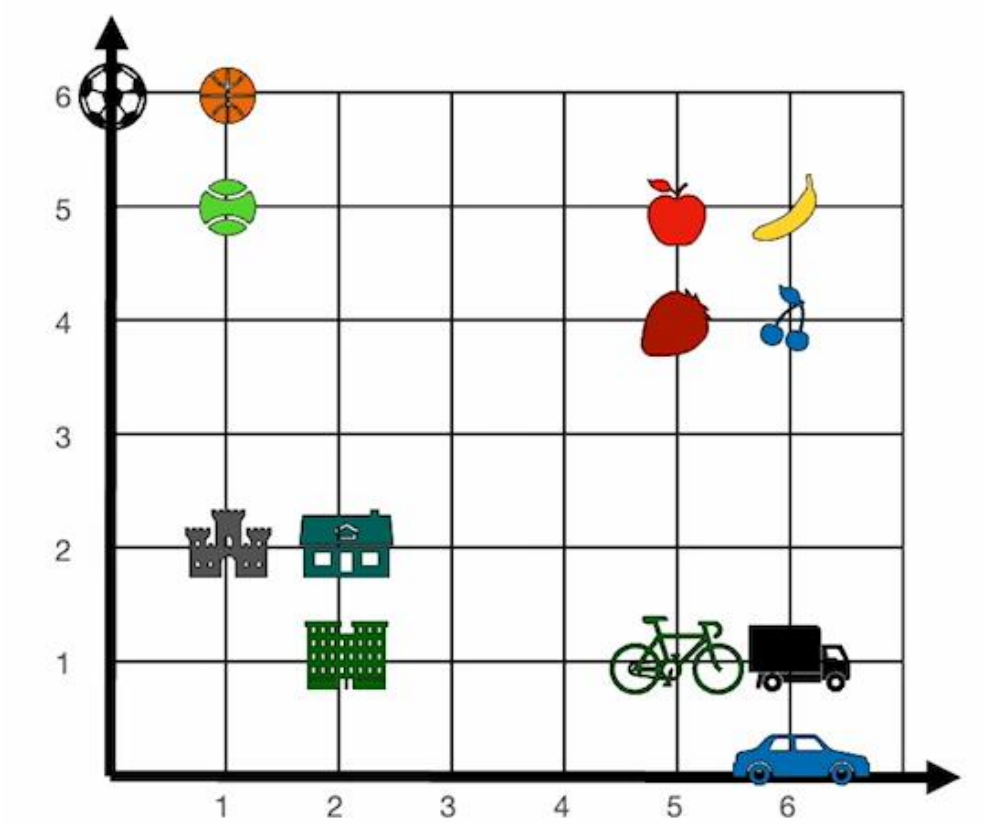
- Maps textual data into a latent space (1024 dimensional).
- Can embed single words, sentences, or entire paragraphs.

Representation:

- Related words or phrases with similar meaning are close in the latent space.

Our application of embedding models:

- Retrieve the most relevant content from our database based on user questions.
- This is done by taking the closest data point(s) to the question in the latent space.
- Utilizing open-source embedding models:
  - Model in use: BERT e5-large-v2*
  - Previous semantic-search models: Word2Vec, GloVe, FastText, …



* Bidirectional Encoder Representations from Transformers. EmbEddings from bidirEctional Encoder rEpresentations (https://arxiv.org/pdf/2212.03533.pdf)

2. The Large Language Model (LLM)

Definition:

- Deep learning models trained on vast amounts of text to understand and generate human-like language.

Capabilities:

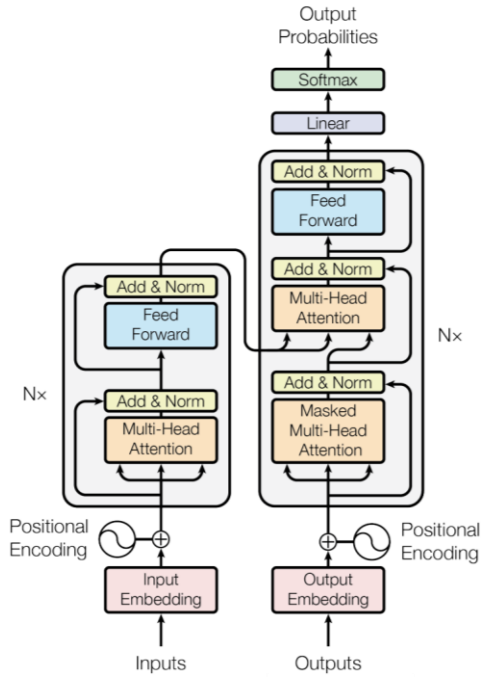
- Deep text understanding and context-awareness.

Application:

- Chatbots.

BERT

Encoder



GPT

Decoder

Our implementation:

- Using pre-trained open-source LLMs: **LLaMA, Mixtral.**
- Future plans:
  - Fine-tune them on our data.
  - Compare them (for our tasks) to commercial models (e.g. GPT4).

MISTRAL AI_

Data Collection: Web Scraping

Purpose:

- Extract source code from web domains and convert to meaningful text strings.

Challenges:

- Time-intensive: The scraper needs to be adapted for each webpage domain individually.

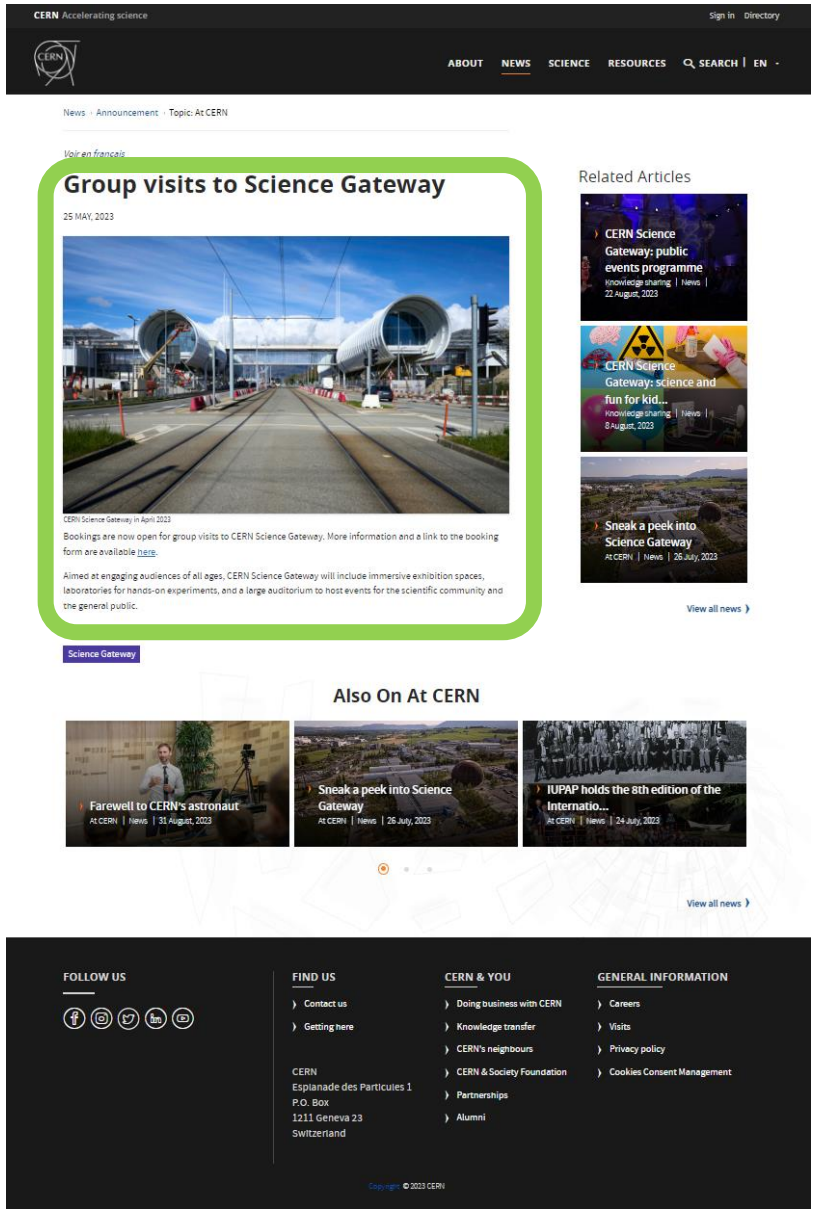
Maintenance:

- Data base has to be updated once in a while.

Advantage of our approach:

- AccGPT knowledge is disentangled from the LLM.

  →No model training necessary for database updates!



“Buy” or “Build”: GenAI Chatbot Solutions

Our Use Cases

Consume Entire Solutions	**Integrate** Models into Custom Frameworks	**Extend** Models via Data Retrieval	**Optimize** Models via Fine-Tuning	**Build** Models From Scratch
Application	Application	Application	Application	Application
Data Retrieval and Prompt Engineering	Data Retrieval and Prompt Engineering	Data Retrieval and Prompt Engineering	Data Retrieval and Prompt Engineering	Data Retrieval and Prompt Engineering
Fine-Tuning	Fine-Tuning	Fine-Tuning	Fine-Tuning	Fine-Tuning
Foundation Model	Foundation Model	Foundation Model	Foundation Model	Foundation Model

3rd Party Managed Custom-Built

→ CERN In-House Chatbot

Key Considerations:

- Highlighted on previous slide.
- In any case, we need to provide inference infrastructure and CERN knowledge.
  - Infrastructure and hardware: Major cost factors.
  - Knowledge preparation: Major time investment.
- Applying data retrieval and fine-tuning LLMs (if GPUs available) not the bottleneck.

Building Our Future: A Strategic Decision. ← CERN critical, moving with "the flow."

- Developing core expertise: Investing in internal capabilities to master LLM and AI chat model technology.
- Staying ahead: (GenAI) Chatbots are transforming how we work → CERN needs to provide chatbots.

→ Goal: Our chatbots will have to be better than others,
but only **for specific CERN use cases.**

Some AccGPT Examples

AccGPT: Examples 1

Disclaimer:
We do not guarantee the accuracy or completeness of the responses provided. We strongly encourage users to verify the information.

How to start a project with AccPy?

To start a project with AccPy, follow these steps:

```
1. Install AccPy: First, you need to install AccPy on your system. You
can do this by sourcing the setup script provided in the AccPy distribution.

2. Create a new project directory: Use the "acc-py init" command to
create a new directory for your project with a suitable name.

3. Set up the project structure: AccPy will automatically set up a
Python package structure in your project directory, including an "__init__.py"
file and a "__main__.py" file.

4. Install dependencies: Use the "acc-py venv" command to create a
virtual environment for your project and install any necessary dependencies using
the "pip" package manager.

5. Write your code: Now you can start writing your code in the
"__main__.py" file or any other file in your project directory.

6. Run your code: To run your code, simply activate your virtual
environment and execute your script using the "python" command.

7. Test and debug: Use AccPy's built-in testing and debugging tools to
ensure that your code is working as expected.

8. Deploy your application: Once you are satisfied with your
application, you can deploy it to a production environment using AccPy's
deployment tools.
```

For more information, you can visit:
https://wikis.cern.ch/display/ACCPY/Getting+started+with+Acc-Py

AccGPT: Examples 2

Could you give an example implementation of an optimization problem in COI?

The example implementation provided is for a linear optimization problem with linear constraints. However, COI also supports nonlinear optimization problems with nonlinear constraints. Here is an example implementation for a nonlinear optimization problem with nonlinear constraints:

```python
from cernml.coi import NonlinearConstraint
from cernml.coi import SingleOptimizable
class MyOptimizable(SingleOptimizable):
    def __init__(self):
        self.constraints = [
            NonlinearConstraint(self._constrain_function, 1e10, np.inf),
        ]
    def compute_single_objective(self, params):
        self._apply_params(params)
        return self._calculate_loss()
    def _constrain_function(self, params):
        self._apply_params(params)
        return self._calculate_function()
```

In this example, `self._constrain_function` is a nonlinear function that must be kept within certain bounds during optimization. The `NonlinearConstraint` class from the COI package is used to specify this constraint for the optimization problem. The `compute_single_objective` method is used to calculate the objective function, and the `self._apply_params` method is used to apply the parameters to the function.

For more information, you can visit: https://acc-py.web.cern.ch/gitlab/geoff/cernml-coi/docs/stable/tutorials/implement-singleoptimizable.html

For such tasks we aim to implement a dedicated coding LLM.

AccGPT: Examples 3

How to Run a Single Python Test?

To run a single Python test at CERN's BE-ICS Industrial Controls context, modify the command line as follows:

1. Set the PYTHONPATH to the relevant test directory (e.g., `/home/lgoralcz/Projects/wccadm/wccadm/scripts/tests`).
2. Use Python 3.6 with the unittest module (-m unittest).
3. Specify the test file and class with the desired test method (e.g., `test_wccadmCreateProjAction.TestCreateProjAction.test_parsing_without_flag_path`).

The complete command:

```
PYTHONPATH=/home/lgoralcz/Projects/wccadm/wccadm/scripts/tests python36 -m unittest
test_wccadmCreateProjAction.TestCreateProjAction.test_parsing_without_flag_path
```

For more information, you can visit:

- [75.9% confidence] https://confluence.cern.ch/display/WOSSIACK/How+to+Run+Single+Python+Test

Feedback

Correct answer!

However, (for me) unclear what tool / framework is meant.

The response is even more detailed than the webpage.

/ BE-ICS Industrial Controls at CERN KB / How-to Articles Analytics

How to Run Single Python Test

Angelegt von Lukasz Goralczyk, zuletzt geändert am Dez 07, 2018

Introduction

How to run Python specific test

Step-by-Step Guide

Use this (modify) command line:

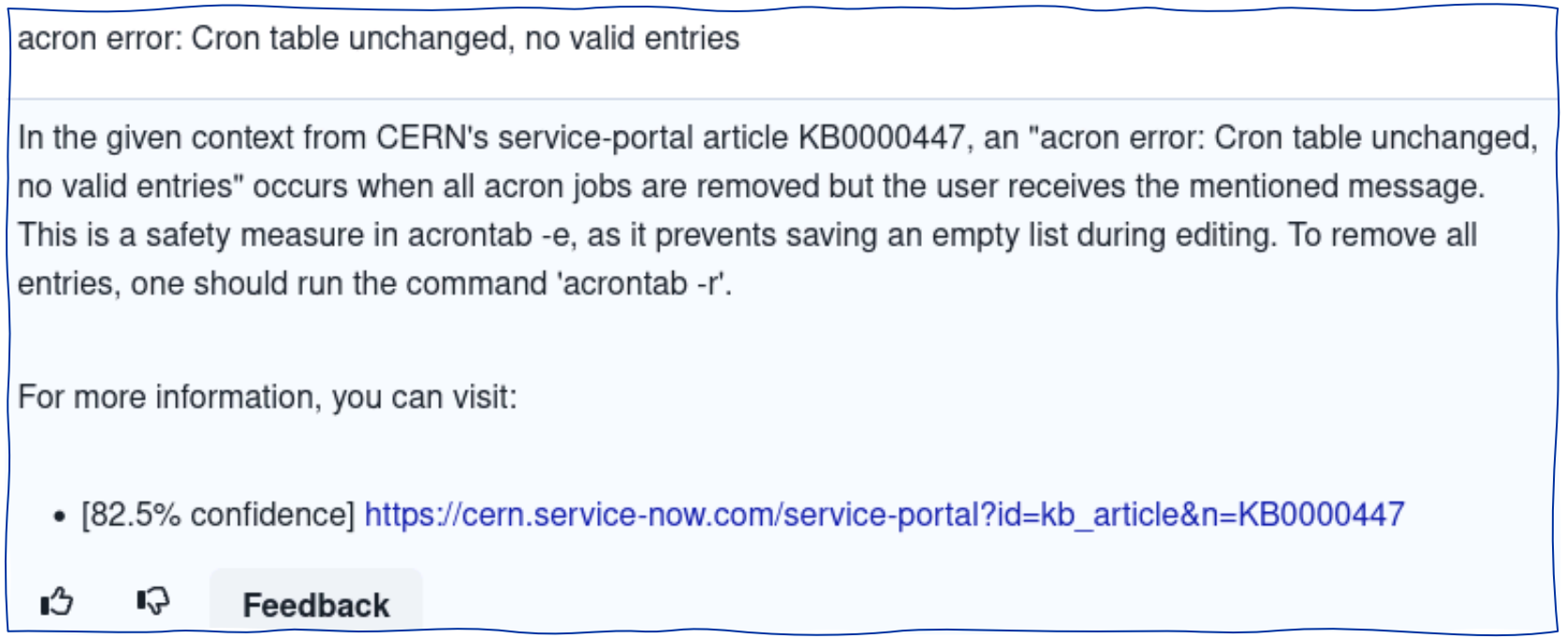
```
PYTHONPATH=/home/lgoralcz/Projects/wccadm/wccadm/scripts/tests python36 -m unittest
```

Related articles

Production vs Development WinCC OA Projects

AccGPT: Examples 4 - SNOW

- **AccGPT can answer frequently asked questions which are documented well.**
  → Can reduce effort to answer support tickets.



AccGPT: Examples 6

What is GPN and TN?

At CERN, GPN (General Purpose Network) and TN (Technical Network) are distinct physical network installations. GPN is primarily used for office computing and general IT services, while TN (which can be interchangeably referred to as EN or Experiment's Network) is dedicated to control systems, online farms, and data acquisition systems for the technical infrastructure and accelerator complex or physics experiments. These two network domains (TN/EN) have similar functionalities but serve different purposes within CERN.

For more information, you can visit:

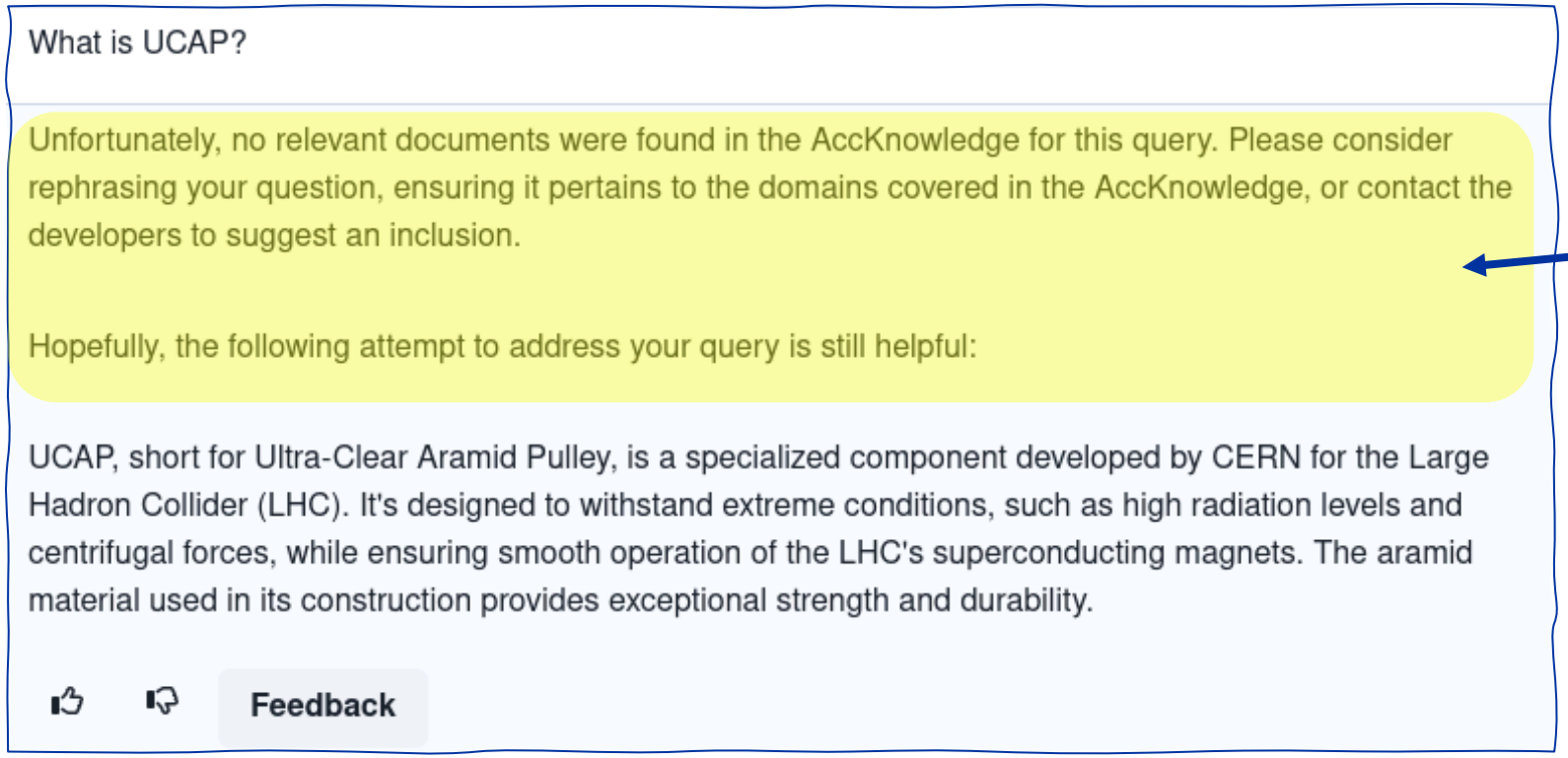
- [58.1% confidence] https://edms.cern.ch/ui/file/584092/2.4/CNIC_Security_Policy_V2.4.pdf

Feedback

AccGPT is on TN!
→ Entirely “offline.”
- Data security is assured.

AccGPT: Example of Not-Included Knowledge

Unified Controls Acquisition and Processing Framework (UCAP)



Text to indicate that this knowledge domain is not included in the AccKnowledge.

Warning that the following response is prone to hallucination.

Data Quality Importance

Not only the AccGPT pipeline determines the model accuracy:

- **Data Quality is KEY !!!**
  (garbage in … garbage out)
- Important aspects:
  - Keep documentations updated.
  - Keep documentations detailed.
  - Duplicates make the model struggling.
  - Text based information is best.
  - Ideally: FAQs and examples.
- For coding:
  - Templates or examples make it easier for the model.

By GPT 4

→ Keep this in mind if you plan to make your documentation accessible to chatbots.

AccGPT: Current Stage

Collaborative Effort Across Departments

Key Contributors:

- IT Department:
  - Task: Infrastructure management and model hosting.
  - Team: (0.5 FTEs) Juan Manuel Guijarro, Sofia Vallecorsa.
- BE Department:
  - Task: Development of the chatbot model.
  - Team: (1 FTE) Florian Rehm, Verena Kain.

Additional Support:

- EN Department (Nathan Soufflet): Provides a user-friendly interface.
- Volunteers: Contributions from some CERN members on a volunteer basis.
  - **Giovanni Guerrieri**, ...

Goal:

- Provide AccGPT for entire CERN.

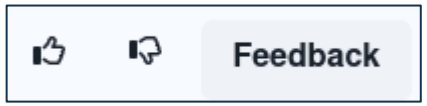
AccGPT: Current Stage

Proof-of-Concept (Knowledge Retrieval): Successful !

- Achieved a promising (good enough) accuracy.

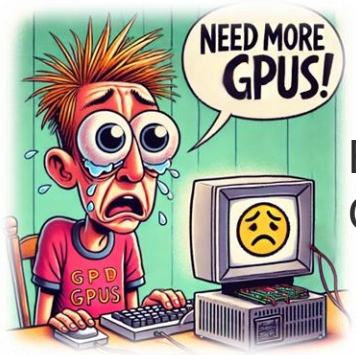
→ Now: Improving the Model.

- Improving the model's accuracy.
- Scaling up: Expanding the model knowledge.
- Enhanced User Interface.
  - Allows feedback collection → excessive testing.



- Testing different open-source models, experimenting with different prompts, …
  - Also, compare accuracy to commercial models.

AccGPT: Next Steps



By GPT4

Need to significantly scale up GPU resources:

- Quality strongly depends on available GPUs: More GPUs allow usage of larger and better models.
  - E.g., the smaller models “hallucinate” often, do not allow a (coherent) chat-like conversation, are very prompt sensitive.

→ Received (temporary) a mini-GPU cluster equipped with 4x A100.

  - Not sufficient to host the “large” open-source models, but a good starting point.

Evaluation:

- Feedback-driven: A community-driven enhancement process for a comprehensive vertical experience.

Further Ideas:

- Fine-tune LLM: Customize with CERN specific data to improve performance and relevance.
- Enhancements to coding tasks:
  - Utilize a dedicated coding foundation model.
- Multimodal expansion: Also consider plots, pictures, videos (presentations, lectures), …

AccGPT: Next Steps - Agents

```
In [5]: agent_executor.invoke({"input": "What is the phone number of Florian Rehm and where is his office"})

> Entering new AgentExecutor chain...

Invoking: `CERN_phonebook_search` with `{'query': 'Florian Rehm'}`

    Preferred name: Florian REHM
    Organic Unit: BE-CSS-DSB
    Email: florian.matthias.rehm@cern.ch
    Phone number: 66134; fullNumber: "+41227666134"
    Office location: 774/2-030
    Mail box: Z10400

> Finished chain.

Out[5]: {'input': 'What is the phone number of Florian Rehm and where is his office',
 'output': "Florian Rehm's phone number is +41227666134 and his office is located at 774/2-030."}
```

- **Agents can do sub-tasks by calling functions.**
- **For example:**
  - Querying the CERN phonebook.
  - However:
    This feature will not come (soon),
    due to data privacy reasons.

Conclusions

Conclusions

- **AccGPT =** a chatbot pilot for CERN specific knowledge retrieval.
- Continuous improvement and knowledge expansion.
- Further extensions (such as coding assistance).
- (So far:) Entirely open-source! We will test the limitations…