SmartAPS: Tool-augmented LLMs for Operations Management

Timothy TL Yu[1], Mahdi Mostajabdaveh[1], Jabo S. Byusa[1], Rindranirina Ramamonjison[1], Giuseppe Carenini[2], Kun Mao[3], Zirui Zhou[1], and Yong Zhang[1]

1. Huawei Technologies Canada
2. University of British Columbia
3. Huawei Cloud Computing Technologies

39th Annual AAAI Conference on Artificial Intelligence

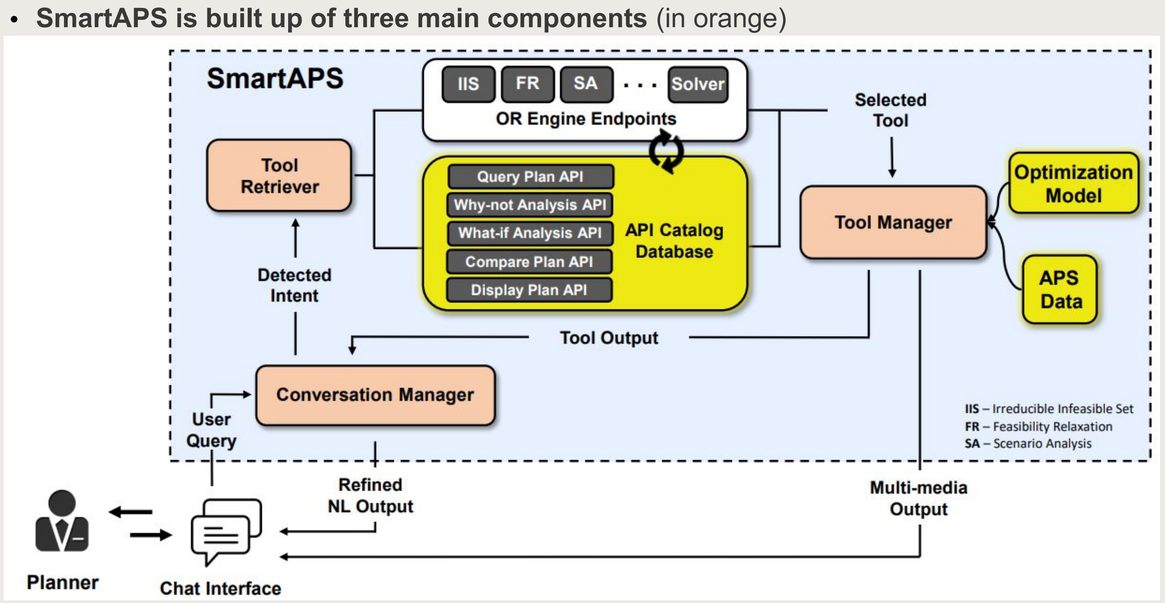
Introduction

- Operations research (OR) is a discipline within applied mathematics that delivers advanced analytical tools to aid in decision-making.
- Advanced planning systems (APSs) are systems developed to aid operations planning and supply chain management. However, their high costs stemming from limited automation, need for customization, and reliance on expert consultants [1] hinder widespread adoption.
- We present SmartAPS, a conversational interface that allows users to perform advanced tasks using natural language.
- SmartAPS leverages the conversation history and tool-augmented generation to describe details about the plan, perform scenario analyses (i.e., what-if and why-not analyses), and plan comparisons.
- A user study was conducted and planners reported that SmartAPS enabled them to query plans and perform analyses more efficiently (from potentially 1-2 days down to a few hours).

SmartAPS Overview

1 https://github.com/Chainlit/chainlit
2 https://python-poetry.org/
3 https://www.trychroma.com/

- **Technology Stack**: We developed our system using a technology stack of **Chainlit[1]** (chat interface), **Python**, **Poetry[2]** (dependency management), and **ChromaDB[3]** (database). Leveraging Chainlit on the client side, a chat interface is used to accept natural language as input to interact with an APS.
- **Models & Solver:**
  - **RAG (tool retrieval): ChromaDB** and **BGE-LARGE-EN-v1.5** [2] & cosine similarity
  - **Solver:** Huawei Cloud's OptVerse AI Solver [3]
  - **LLM:** Mistral-7B-Instruct-v0.1 [4]
- **SmartAPS is built up of three main components** (in orange)



Background

Tool Contract

The contract is used for three purposes: (1) tool retrieval (description, examples, formula), and (2) function calling formula

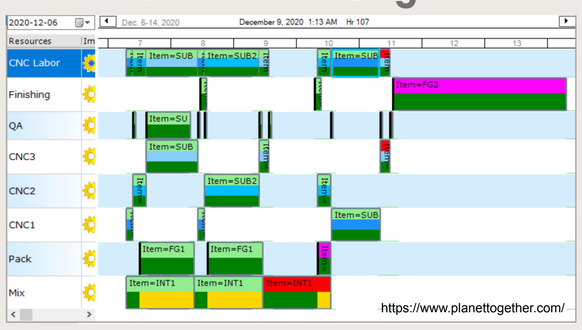
Description	Asking if it is possible to produce some quantity of a specific product by a specific day at any possible plant. Optionally, whether to change the production plan of some other items.
Examples	1. I want to make [ORDER] on time. 2. Why can't we produce [AMOUNT] of [ITEM] by [DATE]? 3. [AMOUNT] of [ITEM] must be made by [DATE] and [RELAXED_ITEM_LIST] may change.
NL Output	Produce [AMOUNT] of [ITEM] by [DATE] allowing production of [RELAXED_ITEM_LIST] to change.
Function Call	produce_before_in_any_plant(output, missing_keys, product, period, amount2produce, order, relaxed_items=None)
Input	`"product": { "type": "str", "label": "ITEM", "required": true }` … `"order": { "type": "str", "label": "ORDER", "required": false }`
Output	`"output":{ "nl_output": "str", "feasible": "bool", "pars2change": "dict", "plan": "nested_dict" }`

Tool Code

Each tool contract is tied to the tool code. When a tool is retrieved, then the corresponding tool code may be

```
def produce_before_in_any_plant(solver_output: SolverOutput, formula:str, product=None, period=None,
amount2produce=None, order=None, relaxed_items=None):
    """ This method will automatically force the production of 'amount2produce' unit of 'item' before
'period' at any plant
        while allow the production of 'relaxed items' change.
    """
    nl_formula = populate_formula(formula, product, period, amount2produce, order, relaxed_items)

    # Create the list of items that we can change their production plan
    relaxed_items = ut.real_item_names_from_list(solver_output.aps, relaxed_items)

    # Fix production of items NOT in relaxed_items
    tg_item_id = solver_output.aps.name_to_id(product, EncodeType.ITEM)
    updated_model = ut.fix_item_production(solver_output.model, exceptions=relaxed_items + [tg_item_id])

    # Add a constraint on production (x variable) of item
    t_id = solver_output.aps.name_to_id(period, EncodeType.PERIOD)
    item_id = solver_output.aps.name_to_id(product, EncodeType.ITEM)

    updated_model.production_const = Constraint(expr= sum(updated_model.x[item_id, p, t]
                                                           for p in solver_output.aps.production[item_id].keys()
                                                           for t in range(solver_output.aps.n_periods) if t <= t_id) >=
amount2produce)

    # Solve and return solution
    updated_solver_output = solve_model(updated_model, solver_output.aps)
    output = ut.analyses_updated_model(nl_formula, solver_output, updated_solver_output)
    return output
```

APS – Floor Scheduling



- Industry leaders include **Kinaxis' Planning One**, **SAP's S/4HANA**, and **Oracle's Fusion Cloud Supply Chain & Planning**
- Moving towards introducing AI; currently these systems **primarily querying information, demand prediction, and abnormality detection[4]** rather than help perform analyses
- Figure shows a screenshot of a scheduling chart of PlanetTogether APS

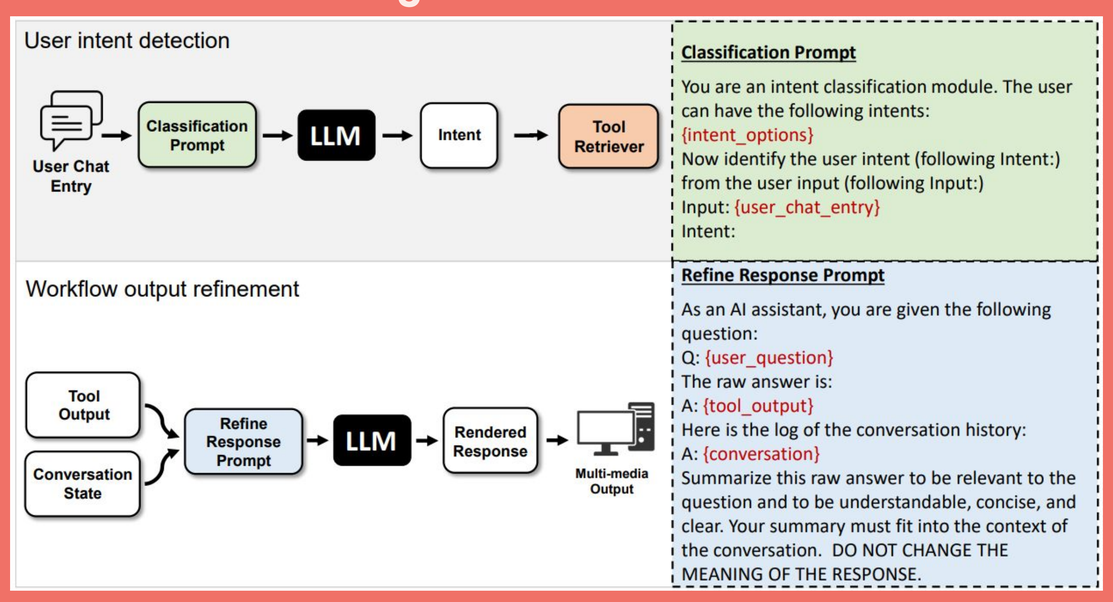
4 https://www.loganconsulting.com/blog/using-ai-to-optimize-demand-planning-with-microsoft-dynamics-365-supply-chain-management/

SmartAPS Details

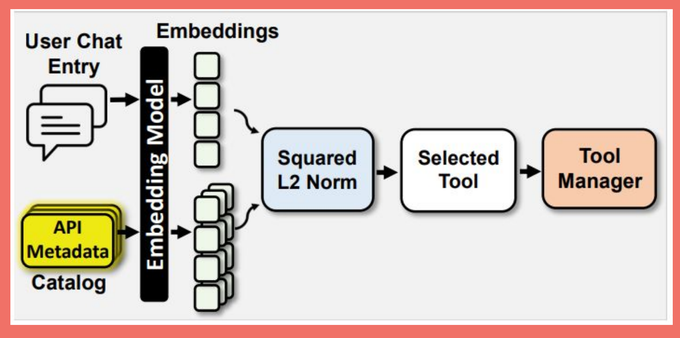
Three main components:

- **Conversation Manager** – two primary tasks: (1) user intent detection, (2) output refinement
- **Tool Retriever** – converts the user query into an embedding vector and then calculates the semantic similarity (cosine similarity in our implementation) between it and the cached embeddings for each Tool API in the ChromaDB collection
- **Tool Manager** – using the tool contract, extracts the input parameters from the user query and the required model & data; executes the tool and returns response

Conversation Manager



Tool Retriever

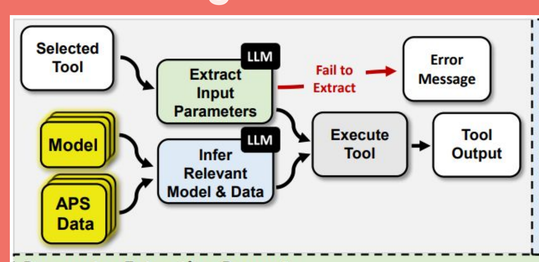


Retrieval Performance

Evaluated on 150 test prompts on 37 APIs created by experts

Tool Categories	# / instances	Retrieval Performance
Query Plan	4	0.938
Why-not	6	0.875
What-if	5	1.000
Compare Plan	16	0.833
Display Plan	6	0.958
Total	37	0.888

Tool Manager



Model Identification Prompt

Based on the conversation, what is the name of the model that the last question from the user is referring to? Respond with the name of the model only.
Q: {conversation}
Output: Based on the conversation, the name of the model is

Parameter Extraction Prompt:

You are a highly intelligent and accurate operations planning domain Named-entity Recognition (NER) system. You take a user question as input and your task is to recognize and extract specific types of operations planning domain named entities in that given passage and classify into a set of following predefined entity types: {entity_types}
Each entity type has values in a particular datatype. The entity type to datatype mapping is given below,
{entity_type_to_datatype_map}
Your output format is only '{{"entities":[{{"label": type of entity from predefined entity types, "value": entity or list of entities in the input text}}]}}'
Input: {text_input}
Here is the conversation for context: {conversation}
Output:

Case Study / Demo

Discussions with Industrial Supply Chain Planners

Through our discussions with Huawei's supply chain planners and observations of their workflows, we identified that the most common types of analyses required by planners include finding reasons for customer order production delays and identify resolutions. To answer production planners needs with SmartAPS, we asked OR consultants to develop tools with APIs and API contracts specifically designed for production planning. The tool categories and the number of instances for each category are presented in the table in "retrieval performance".

Developed Tools used in Demonstration

The tools that were developed by the OR consultants are used in the presented demonstration. The ones of particular interest and value were those that performed scenario analyses (i.e., what-if and why-not analyses). These directly answer the important questions that were asked by the supply chain regarding production delays and identifying resolutions.

Feedback from Production Planners and OR Consultants

Users reported that SmartAPS enabled them to query plans more efficiently and more readily identify the reasons for customers order production delays. They particularly highlighted the advantages of supporting 'why-not' and 'what-if' analyses, which could reduce the time required for analysis from potentially 1-2 days.

Future Work

- Some sophisticated code generation methods should be investigated for their ability to automatically create these advanced APIs with complex algorithms.
- Due to the potential long solve time of optimization solvers, a task manager should be incorporated into SMARTAPS to log and allow tools to be run in parallel.
- Finally, the conversation is currently between the system and one user. In real-world operations, it is common to have multiple planners each with multiple different objectives. Multi-user approaches should be explored.

References

[1] Jiafu Wan, Xiaomin Li, Hong-Ning Dai, Andrew Kusiak, Miguel Martínez-García, and Di Li. 2021. Artificial-intelligence-driven customized manufacturing factory: Key technologies, applications, and challenges. Proceedings of the IEEE, 109(4):377– 398. 9(4):377–398.

[2] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.

[3] Xijun Li, Fangzhou Zhu, Hui-Ling Zhen, et al. 2024. Machine learning insides optverse ai solver: Design principles and applications. arXiv preprint arXiv:2401.05960.

[4] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, et al.. 2023. Mistral 7b.