9th International Workshop on Smart Computing and Applications (SCA)

Supporting Context-aware AI-augmented Data Extraction Features for Industrial Technical Documents

Ngoc Nhu Trang Nguyen
Daienso Lab, Vietnam
nhutrang.nguyen@daienso.com

Hong-Linh Truong
Aalto University, Finland
linh.truong@aalto.fi

09.07.2026

Content is available under
CC BY-SA 4.0 unless otherwise stated

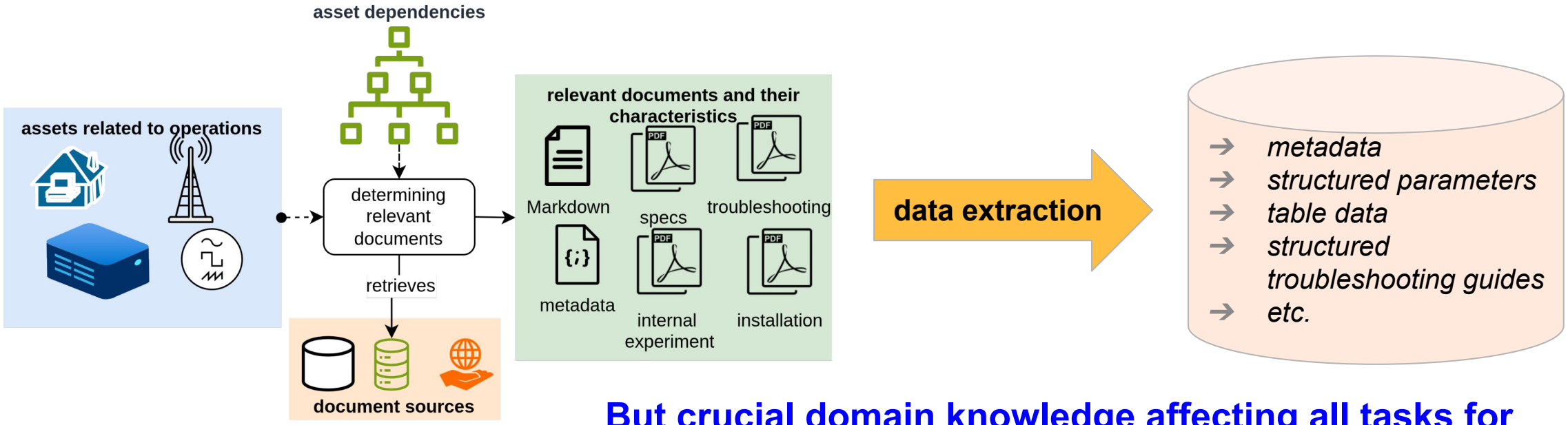
Content

- Motivation
- Context-aware and composable methods for data extraction within AI resource-constrained environments
- Prototype
- Experiments
  - Extraction of telco technical documents
- Conclusion

Motivating example: the case of “knowledge acquisition” for telco operations

Diverse types of technical documents of complex assets and systems

<u>Knowledge acquisition:</u> extracted results as *data products*, which are entered into knowledge sources for analytics and operations.



But crucial domain knowledge affecting all tasks for the robustness, quality and compliance of extraction!

Key requirements for smart tooling

- **Contextual extraction pipelines for balancing cost, efficiency, and regulation**
  - Various extraction situations/goals →different extraction pipelines and results
    - *e.g., installation instructions, troubleshooting instructions, reference data for analytics*.
  - AI resource-constrained environments with edge GenAI/LLMs infrastructures and **non-AI engineers**
  - On-premise implementation and deployment due to data sovereignty and regulations
- **Hybrid intelligence configurations (AI-tool-human) for quality assurance**
  - Various domain quality constraints for different extraction results as data products
  - Many extraction results can only be examined by engineers as SME (Subject-Master Expert), *e.g. troubleshooting instructions.*

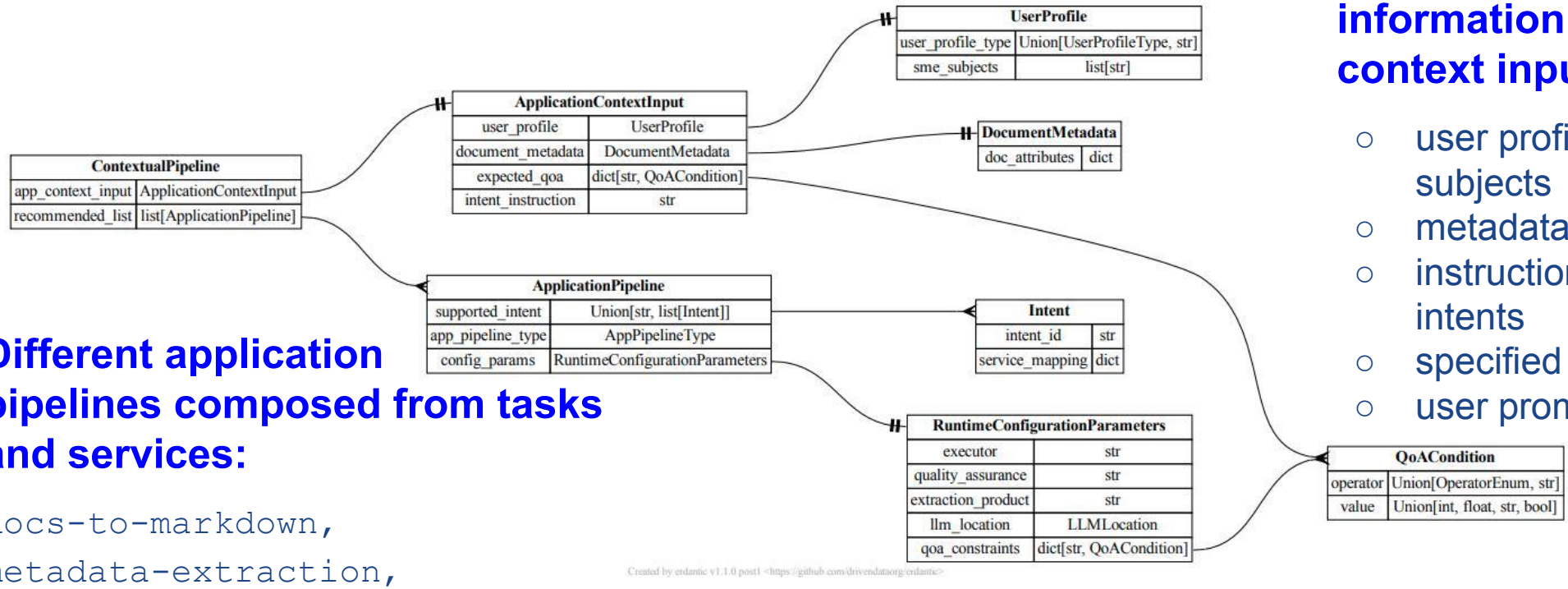
Understanding and modeling attributes of the application context

- **Application context**: situations in which the extraction is carried out from the domain-specific viewpoints
- **Modeling attributes of application context**: capture contextual information from user activities
  - grouped based on context concept of `What` and `Where`
  - propagate into different software components to support context-aware pipeline execution

Type: Attribute Name and Values	
What: executor = { nonllms, llms, agents, hybrid}	control the selection of backend LLMs
What: quality_assurance = {nonllms, llms, sme, hybrid, skip}	address appropriate pipeline configuration
What: user_profile_type = {deployment, operation, optimization, support}; sme_subjects = {topic1, topic2, ...}	tailor configuration parameters to the user's role and assign SME for quality assurance
What: extraction_product = {metadata, table, figure, graph, structured-parameter, structured-guide}	address appropriate quality constraints and configuration of quality assurance
Where: llm_location = {on-premise, cloud, hybrid}	select appropriate LLMs according to access controls, costs, and risks

Examples of some important attributes

Application context inputs and application pipelines for configuring contextual pipelines



Different sources of information for application context inputs:

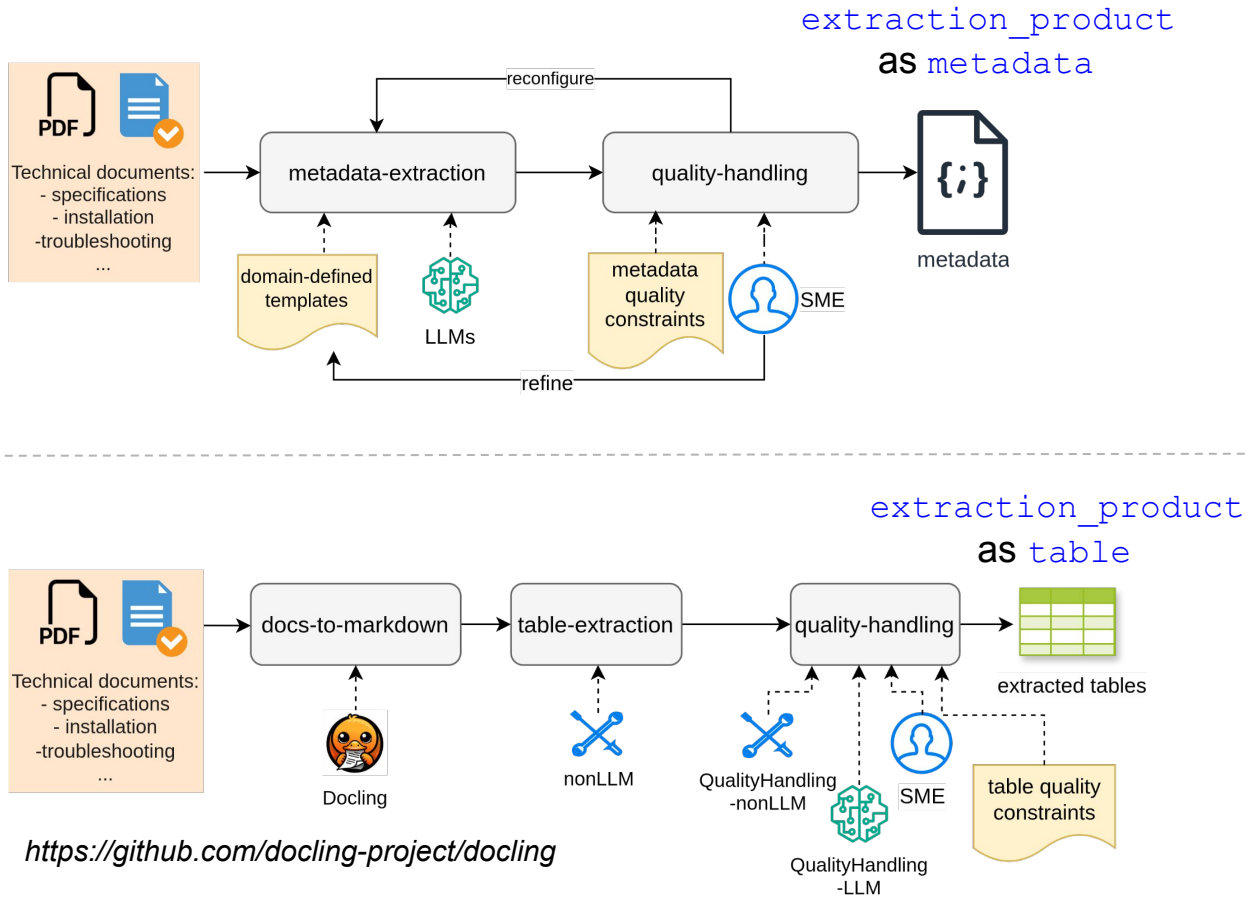
- user profile and SME subjects
- metadata of tech docs
- instruction of extraction intents
- specified expected quality
- user prompts

Different application pipelines composed from tasks and services:

`docs-to-markdown`,
`metadata-extraction`,
`parameter-extraction`,
`table-extraction`,
`procedure-extraction`,
`quality-handling`, etc.

Composing application pipelines

- Composition techniques for existing software components:
  - a concrete extraction task can have different patterns based on the composition of ML/LLMs models
  - a quality control can use different control patterns built atop LLMs and human experts (SME)
- From the pipeline possibilities, possible runtime configuration parameters can be used to control underlying executions

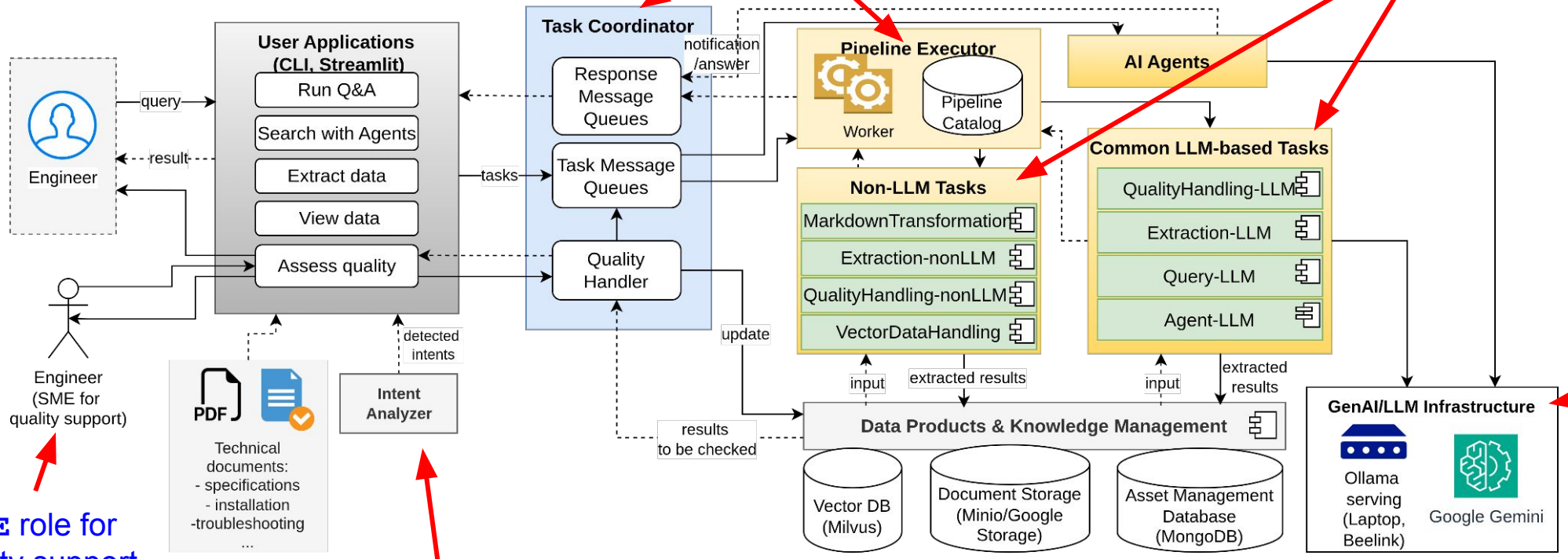


Examples of composing application pipelines with context-aware and quality handling

A high level view of implementation

Task Coordinator interacts with **Pipeline Executor** to perform various extraction tasks

Different **Non-LLM Tasks** and **Common LLM-based Tasks** are based on open sources tools/LLMs and custom software components



SME role for quality support

analysis of intent

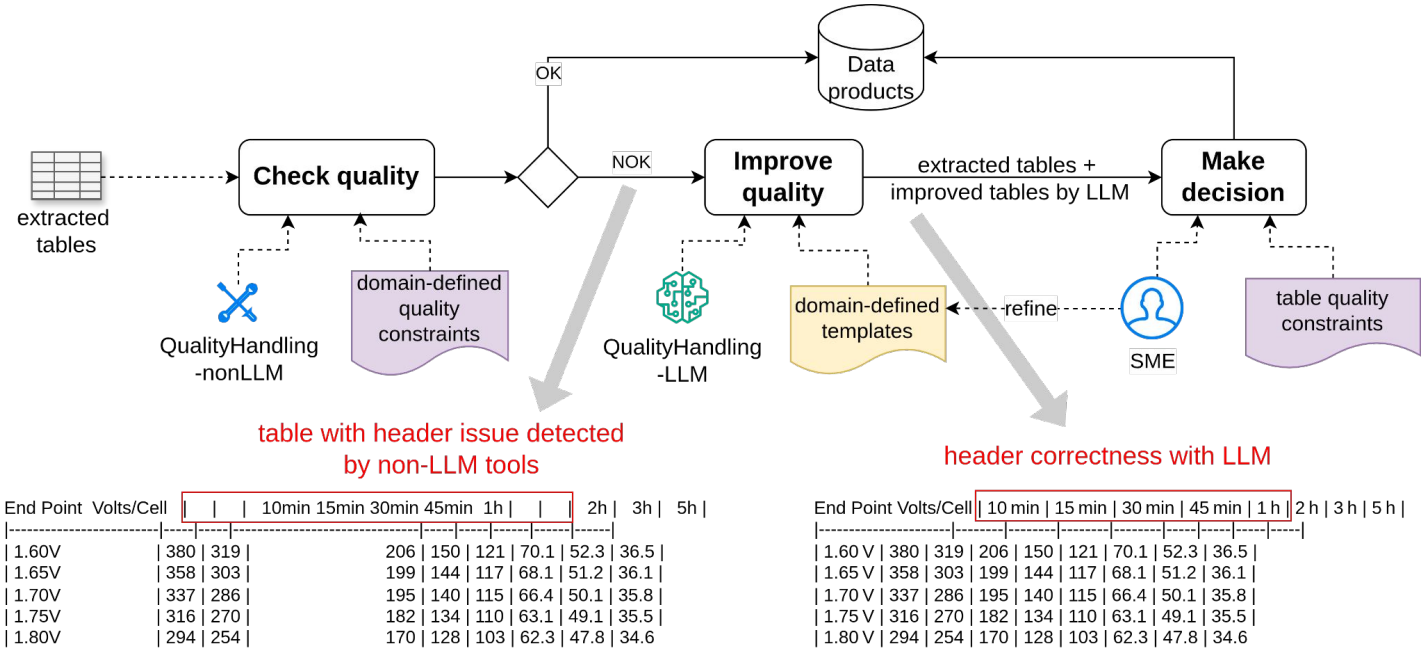
on-premise GenAI/LLMs with public LLMs deployed on edge devices using `Ollama` *(https://github.com/ollama/ollama)*

Experiments with table extraction

Observation of extraction quality:

- **`LLM-based Tasks`** face common problems:
  - cannot identify and extract tables, especially in cases of not well-structured tables
  - extract graphs with grid backgrounds as tables
- **`Non-LLM Tasks`** based on customized code utilities can extract most of tables

see details



End Point	Volts/Cell			10min	15min	30min	45min	1h			2h	3h	5h
1.60V	380	319				206	150	121	70.1	52.3	36.5		
1.65V	358	303				199	144	117	68.1	51.2	36.1		
1.70V	337	286				195	140	115	66.4	50.1	35.8		
1.75V	316	270				182	134	110	63.1	49.1	35.5		
1.80V	294	254				170	128	103	62.3	47.8	34.6		

End Point	Volts/Cell	10 min	15 min	30 min	45 min	1 h	2 h	3 h	5 h
1.60 V	380	319	206	150	121	70.1	52.3	36.5	
1.65 V	358	303	199	144	117	68.1	51.2	36.1	
1.70 V	337	286	195	140	115	66.4	50.1	35.8	
1.75 V	316	270	182	134	110	63.1	49.1	35.5	
1.80 V	294	254	170	128	103	62.3	47.8	34.6	

Contextual quality handling:

- **`Custom tools`** for basic quality checks and ***domain-defined quality*** checks
- **`LLM-based Tasks`** for improving quality of extracted tables

Experiments with troubleshooting procedures extraction

Extraction quality:

- **`Intent Analyzer`** help *enrich user prompt via domain knowledge*
  - e.g., "`VSWR`" is highly correlated with "`antenna_system`"
- **`LLM-based Tasks`** for troubleshooting procedures extraction: *structured domain-defined templates* to guide the extraction
  - assure structural and comprehensive extracted procedures
  - for knowledge acquisition management

Context-assisted quality handling:

- since there is no way to automatically verify the correctness of extracted troubleshooting procedures, **appropriate `SME` for quality handling can be identified using extraction intent and user profile.**

An enriched prompt:

[user prompt]: ... to extract instructions for solving Poor VSWR from ...
[intent enrichment]: ## in system components: "antenna system"
[session_prompt]: ... to follow schema format definition ...

A structured troubleshooting procedure:

```
{
   "faultName":"Poor VSWR",
   "faultComponent":"Antenna System",
   "faultDescription":"An elevated Voltage
   Standing Wave Ratio indicates excessive signal
   ...",
   "commonCauses":[
      "Loose or improperly installed connectors",
      ...
   ],
   "dependencyComponents":[
       "Connectors", "Cable", ...
   ],
   "alarms":[
       "VSWR Alarm", "Return Loss alarm ..."
   ],
   "troubleshootDetails":[
       ...
   ]
}
```

Conclusion

- We present context-awareness and composability techniques for smart data extraction pipelines
  - supporting the incorporation of domain-specific application contexts
  - composing and adapting of application pipelines based on context-awareness
  - dealing with resource-constrained environments (AI infrastructures and AI human resources) and diverse extraction quality controls
- Future work
  - experiment our works and edge GenAI/LLMs with other extraction situations/goals, domains, and large datasets.

Kiitos
aalto.fi