AI for Data Management (aiDM) Workshop @ SIGMOD
May 31st 2026, Bangaluru, India

https://datapact.eu/

VCL - Verbal Command Language:

Bridging Natural Language and Structured Control for Domain-Specific Document Analysis

Renzo Alva Principe * · Filippo Armani * · Matteo Palmonari * · Carlo Batini **

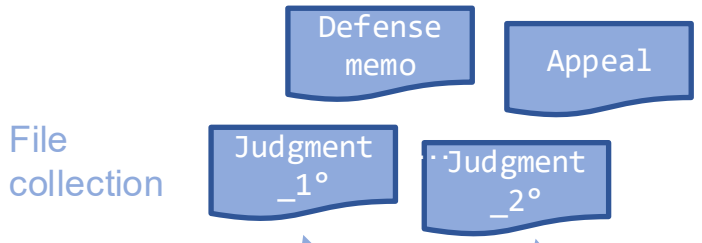
* University of Milano-Bicocca — Department of Informatics, Systems and Communication (DISCo)
** Consorzio Interuniversitario Italiano di Informatica (CINI)

*The work has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101189771 - **DataPACT**, and from the **AI-PACT project** (B47H22004460001), funded by the Italian Ministry of Enterprises and Made in Italy under the European Union Next Generation EU programme and the Italian PNRR. Carlo Batini wishes to express his gratitude to the judges of the Court of appeal of Milan and the Court of Catania, and to the members of the Bar Association of Catania for their valuable collaboration in the experimental evaluation of the VCL language.*

How do experts actually want to use GenAI?

We shadowed judges drafting their opinions over real case files

Beyond question answering

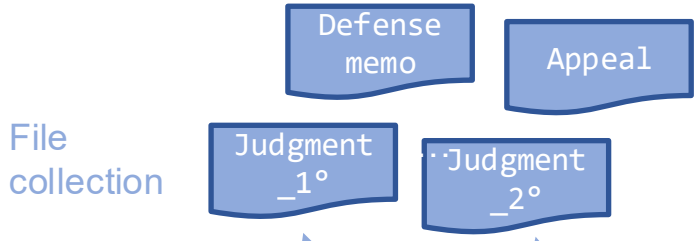


Example of judge's request in a civil appeal case

"Identify the main decisions in the first instance and second instance ruling, and compare them in a bullet point list."

How do experts actually want to use GenAI?

We shadowed judges drafting their opinions over real case files



Beyond question answering

- **Recurring operations** modeled around **speech acts**

  EXTRACT decisions **SUMMARIZE**

  COMPARE rulings **INTEGRATE** results **SEARCH** entities

- **References** to specific documents
- Often, **implicit composition** of different operations

Example of judge's request in a civil appeal case

"Identify the main decisions in the first instance and second instance ruling, and compare them in a bullet point list."

How do experts actually want to use GenAI?

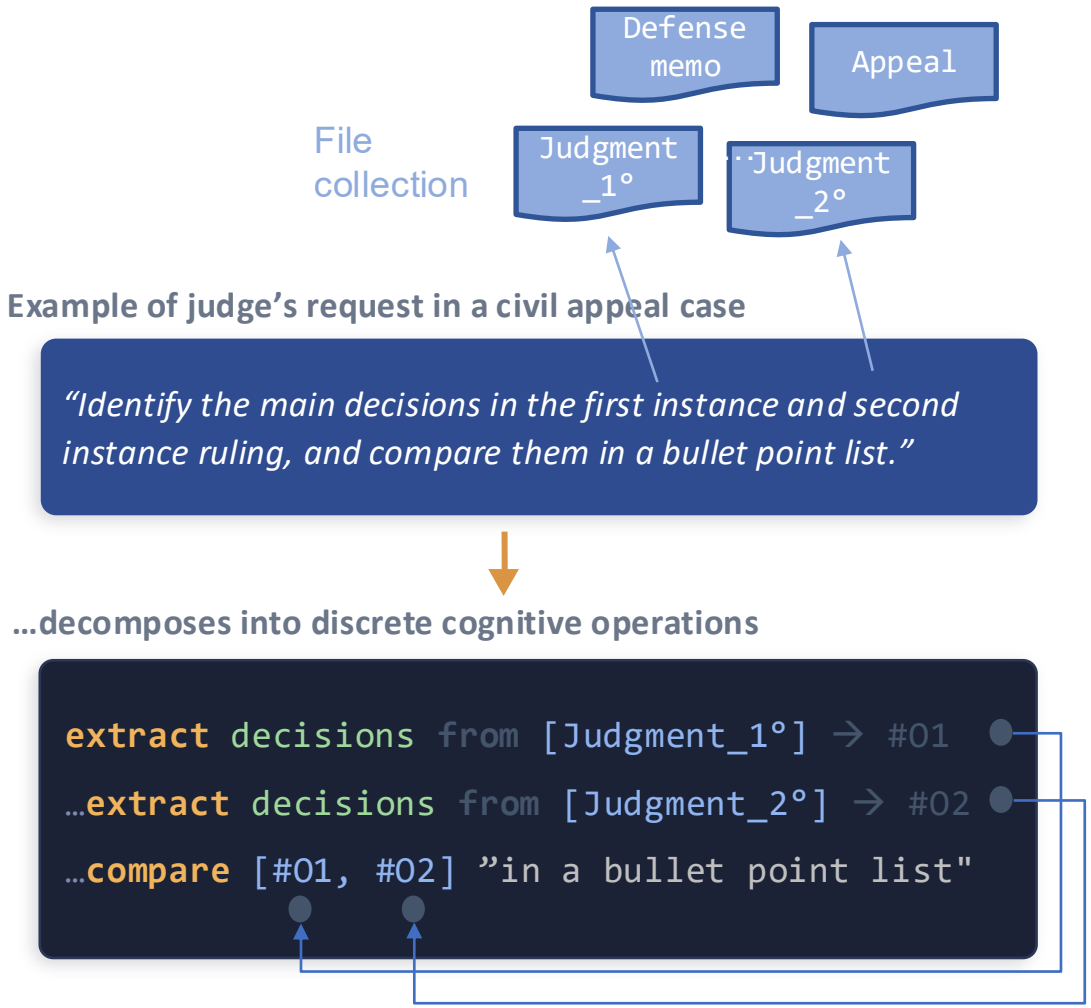
We shadowed judges drafting their opinions over real case files

Beyond question answering

- **Recurring operations** modeled around **speech acts**

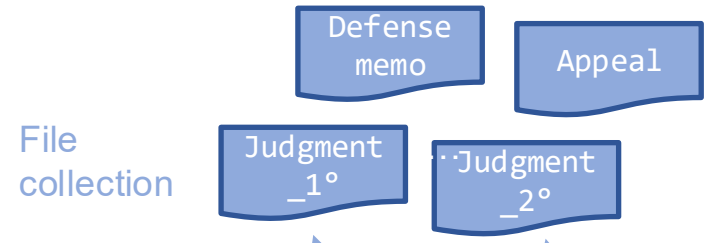
  EXTRACT decisions **SUMMARIZE**
  COMPARE rulings **INTEGRATE** results **SEARCH** entities

- **References** to specific documents
- Often, **implicit composition** of different operations



How do experts actually want to use GenAI?

We shadowed judges drafting their opinions over real case files



Beyond question answering

- **Recurring operations** modeled around **speech acts**

  EXTRACT decisions **SUMMARIZE**
  COMPARE rulings **INTEGRATE** results **SEARCH** entities

- **References** to specific documents
- Often, **implicit composition** of different operations

Example of judge's request in a civil appeal case

"Identify the main decisions in the first instance and second instance ruling, and compare them in a bullet point list."

…decomposes into discrete cognitive operations

```
extract decisions from [Judgment_1°] → #01
…extract decisions from [Judgment_2°] → #02
…compare [#01, #02] ”in a bullet point list"
```

Composition of data manipulation operations on the input files … can we get inspiration from SQL?

From cognitive patterns to a query language for documents

1 Field investigation

- Empirical study with judges
- Observation & think-aloud
- Identify recurring cognitive patterns

2 Conceptualization

- **Speech Act Theory**
- Cognitive activities → verbal commands
- Each command = an intent on a document

3 Verbal command language (VCL)

- A declarative language for analysis
- **Inspired by SQL**
- Document-in / document-out model

4 From VCL to usage

- Prompt engineering methodology
- **Agentic implementation with custom processing**

Key contributions

- Formalization of VCL: **VCL-Query** + **VCL-Definitions**
- Agentic implementation: **VCL Engine**
- **Evaluation** based on a real legal use case

Where VCL sits in the landscape

Unstructured Document Analytics (UDA)

- GALOIS, QUEST, UQE: declarative SQL-like
- LOTUS, DocETL: imperative dataflow
- SQL / code interfaces for data engineers
- Operation sets replicate DB primitives on structured outputs rather than cognitive operations on documents

Query rewriting

- Bridges user intent and system processing
- RaFe: multi-agent rewriting from retrieval feedback
- Agent4Ranking: agentic query rewriters
- Rewriting is not based on domain-specific grounding or abstract modeling

Reasoning in RAG

- SEER, Toolformer, IRCoT, Tree-of-Thoughts, Think-on-Graph
- Interleave retrieval and inference
- Growing sophistication in orchestration
- Reasoning delegated to model → opaque

VCL

- complements UDA with a **natural language interpretation** and **cognitively grounded document-specific operators**
- draws on query rewriting to decompose NL into domain-grounded tasks but **using a target abstraction**
- transform opaque reasoning into **explicit, transparent and traceable plans, operations and outputs**

VCL-Query: a declarative, SQL-inspired language

QUERY STRUCTURE

```
COMMAND [from] what ”conditions”
```

Example:

```
SEARCH [Judgment 1] decision ”report also motivations"
```

COMMAND
the cognitive activity to perform

from
source: documents | results of prior requests

what
target category of elements (entities, sections, concepts)

conditions
modifiers, filters, formatting conditions, etc.

MAIN COMMANDS

SEARCH	find concepts and their instances
EXTRACT	extract portions of text matching a specified category
TRACE	identify logical dependencies (e.g. a timeline)
SUMMARIZE	compress an input text
COMPARE	analyze similarities and differences
INTEGRATE	merge the content of multiple input texts

VCL-Query: grammar

QUERY STRUCTURE

```
COMMAND [from] what ”conditions”
```

Example:

```
SEARCH [Judgment 1] decision ”report also motivations"
```

GRAMMAR

<query> ::= <SEARCH> | <EXTRACT> | <TRACE> |
<SUMMARIZE> | <COMPARE> | <INTEGRATE>
<SEARCH> ::= <from> <what> <condition>+
<EXTRACT> ::= <from> <what> <condition>+
<TRACE> ::= <from> <what> <condition>+
<SUMMARIZE> ::= <from> <condition>+
<COMPARE> ::= <from> <from>+ <condition>+
<INTEGRATE> ::= <from> <from>+ <condition>+
<from> ::= <source> | <query>
<source> ::= VCL-D_{source}
<what> ::= VCL-D_{what}
<condition> ::= <limit> | str
<limit> ::= <sign> <number> <unit>
<sign> ::= ≤|<|≥|>|~| %
<number> ::= $\mathbb{N}^+$
<unit> ::= "word" | "phrase" | "chars"

VCL-Query: domain specialization via VCL-Definitions

QUERY STRUCTURE

```
COMMAND [from] what ”conditions”
```

Example:

```
SEARCH [Judgment 1] decision ”report also motivations"
```

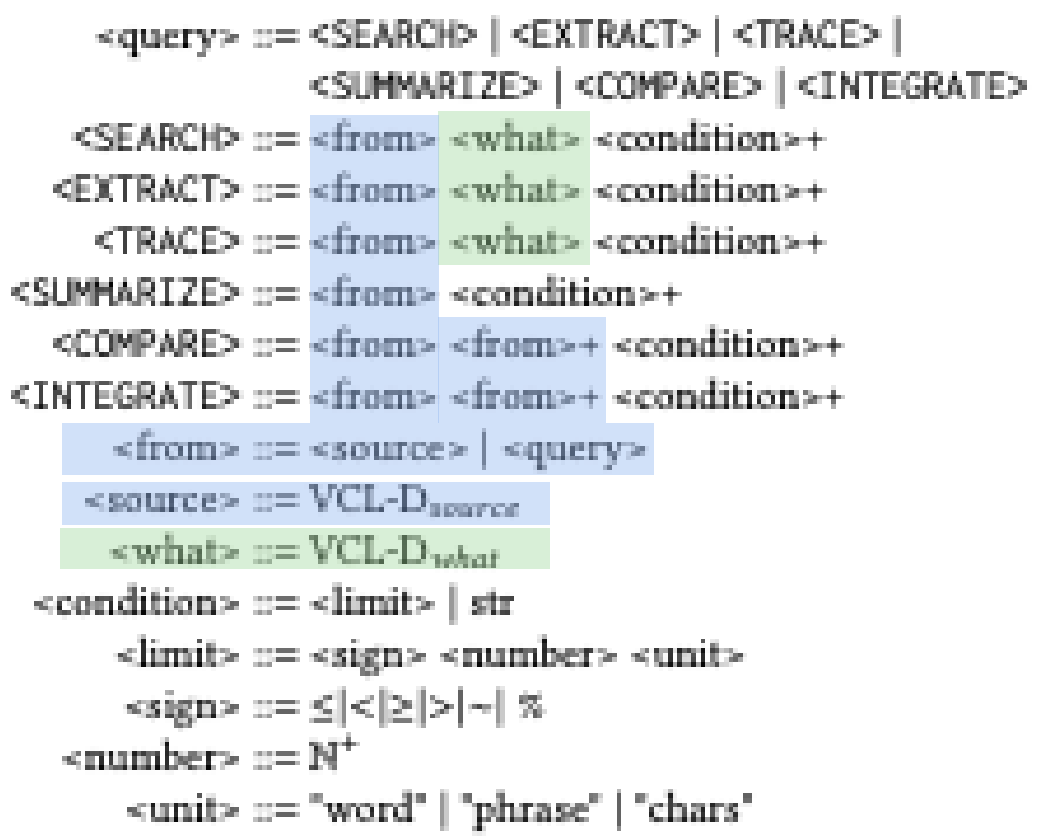
DOMAIN SPECIALIZATION

- References to files may be implicit, use aliases, etc.
- Some commands imply classification operations: explicit definitions improve LLM-based classification

VCL-Definitions

- **Files descriptions** supporting reference disambiguation
- **Domain-specific definitions** of categories, constraints, etc.

GRAMMAR



VCL-Definitions: teaching VCL the language of your domain

VCL-D defines how **user-specified elements** are interpreted and constrained — manually crafted prior to deployment, reused across queries

VCL-D_source

Example for Judgement_1° – a court ruling

```
label          Judgment_1°
synonyms       "first-instance ruling"
description    "the ruling issued by the trial
                court"
```

VCL-D_what

Example for "decision"

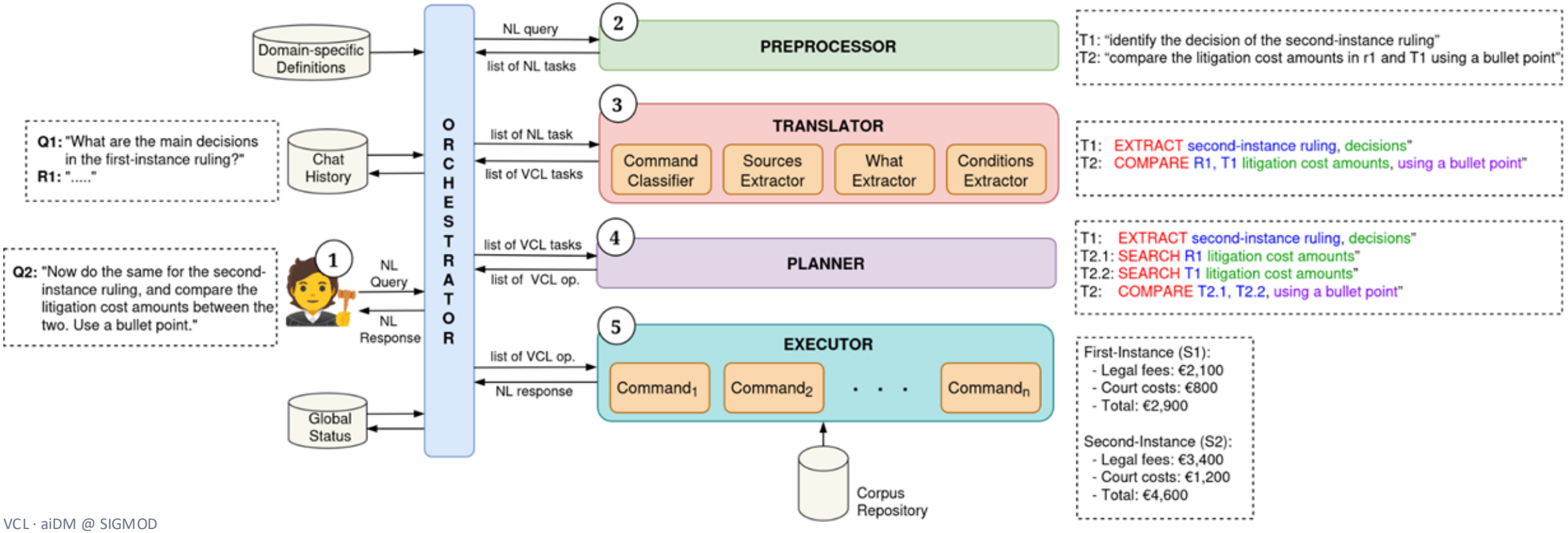
```
name           decision
description    "the operative part of a
                ruling …"
availability   [Judgment_1°, Judgment_2°]
```

Why user-defined? adapts to any domain *without modifying the core engine* · knowledge base **explicit · auditable · controllable**

Agentic implementation: VCL Engine

Key principles:

- From **natural language** prompts **to VCL-based task decomposition and execution**
- **Specialized agents** for processing steps and command execution
- Pipeline-based **orchestration** with LangChain + MongoDB with **full log of operations and data persistence**



EVALUATION

Experimental setup

Goal: compare output quality of explicit task structuring in VCL vs. alternatives along three axes: (1) **retrieval strategy**, (2) **LLM size** and (3) **reasoning capabilities**

Dataset

- one real court case
- 4 anonymized docs
- 36 queries with gold answers (mixed complexity)

Approaches / models *

- NotebookLM (Gemini 3.0)
- Copilot (GPT-5.1)
- Naive RAG (GPT4o-mini | GPT-5.1)
- OpenAI FileSearch (GPT4o-mini | GPT-5.1)
- **VCL (ours)** (GPT4o-mini | GPT-5.1)

Metrics

- Precision
- Recall
- F1-score
- Faithfulness

Verification

- LLM-as-a-judge
- decomposition of outputs into claims
- supported / not supported

* large reasoning models are underlined

LIMITATIONS

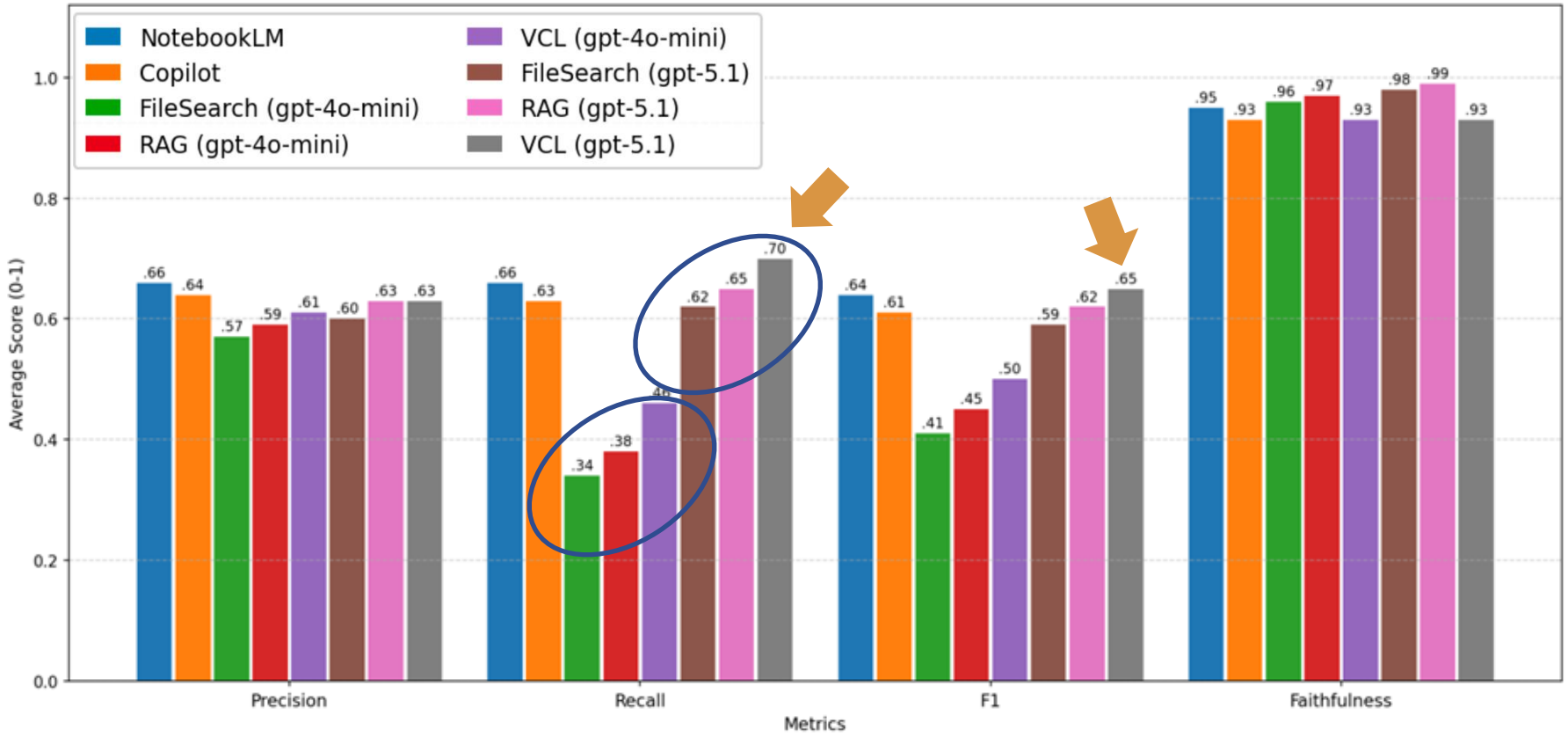
Small dataset · non-expert annotators · subjectivity of some commands · VCL may shape question phrasing · judge-LLM capability (especially: verbose irrelevant but not false claims are penalized → recall more important).

Results & key findings

Ours:

VCL gpt-4o-mini

VCL gpt-5.1



Competitive — and best in Recall: VCL (gpt-5.1) leads on Recall (0.70) and F1 (0.65), on par with commercial assistants.

Controlled retrieval ↑ Precision: VCL queries only the documents named in the request; RAG / FileSearch lose document boundaries at indexing and over-retrieve.

Guided execution enforce constraints: When the corpus doesn't support the request, VCL returns "Not possible" — other models still answer.

CONCLUSIONS

What we found — and what's next

MAIN FINDINGS

Competitive performance
Formal language-based planning achieves promising results — excelling in Recall

Context control
VCL pre-selects the relevant documents, guaranteeing a focused, scoped context

Rigidity ↔ flexibility
Strict adherence to the plan can sometimes be counter-productive

ONGOING / FUTURE WORK

1 Evaluation

- Comparison with UDA approaches
- Field experiments of VCL-based methodologies with judges and layers

2 Extensions

- Tabular data extraction – from UDA comparison
- Command extension – from field experiments

3 Optimization and integration

- Optimization of VCL Engine along multiple aspects
- Integration with DAVE interface [IJCAI25] and constrained generation [ISWC25]

DATAPACT
https://datapact.eu/

Questions?

VCL — Verbal Command Language · document query language for domain users

Code is open source: https://github.com/unimib-datAI/VCL
If you are interested in a demo, get in touch: matteo.palmonari@unimib.it

EXAMPLE FROM PROTOTYPE USER INTERFACE

Example: transparency and traceability

▼ Task 1: Compare

The Task was represented in DQL as follows:

```
{
    "command": "confronta",
    "from": [
        "sentenza di tribunale",
        "sentenza di corte d'appello"
    ],
    "what": [
        {
            "name": "dispositivo"
        }
    ]
}
```

Interpretation in VCL

After being translated, it was necessary to break the task down into 3 operations.

Since the intent is “compare”, the task is decomposed in three operations

▼ Operation 0: Extract Semantic

The single operation command is:

```
{
    "command": "estrai semantico",
    "from": [
        "sentenza di tribunale"
    ],
    "what": [
        {
            "name": "dispositivo"
        }
    ]
}
```

Operation 0: EXTRACT
- Formalization
- Outoput

▼ View the result of the operation

Analysis of the Judgment's Operative Part Context of the Decision The judgment in question concerns an appeal filed by Patrizio Gini, Luigi Mori, and Laura Voni against Telelink Italia SpA for the determination of the illegitimacy of absorptions and reductions of the "individual premium" starting from February 2018. The appellants request payment of the sums due and the recalculation of the remuneration institutions. **Judge's Decision** The judge, after examining the parties' requests and the documentary evidence, upheld the appeal, declaring the illegitimacy of the absorptions made by Telelink Italia SpA starting from February 2018. *Sentences Imposed* 1.

Comparison between the Court of First Instance ruling and the Court of Appeal ruling. Similarities

1. **Subject of the Dispute** : Both documents deal with issues relating to the absorption of salary sums, in particular the "individual surcharge" and the "superminimum" salaries, involving the same company, Telelink Italia SpA, and the same appellants (Gini and Mori) and a third party (Voni).
2. **Final Decision** : Both proceedings conclude with a decision that implies an obligation to pay and recalculate the sums owed, although with different outcomes for the various appellants.
3. **Regulatory References** : Both documents refer to legal principles regarding wage absorption and corporate use, highlighting the importance of these concepts in the context of decisions. Differences
4. **Outcome for Laura Voni** :
   - **Court Ruling** : The ruling upholds Laura Voni's appeal, ordering payment of the sums owed and a recalculation of the wage institutions.
   - **Court of Appeal Ruling** : The court dismisses the dispute against Laura Voni, awarding costs, without considering the merits of the case.
5. **Decision on the Appellants Gini and Mori** :
   - **Court ruling** : Accepts the appeal and orders Telelink Italia SpA to restore the individual surcharge and pay the sums due.
   - **Court of Appeal Ruling** : Rejects Telelink Italia SpA's appeal and upholds the first-instance ruling, deeming the grounds
6. **Lite expenses** :
   - **Court ruling** : Telelink Italia SpA is ordered to pay legal costs totaling €3,200.00.
   - **Court of Appeal ruling** : Telelink Italia SpA is ordered to pay legal costs totaling €2,200.00, specifying that the costs are to be paid to the workers' lawyers.
7. **Business Use References** :
   - **Court Ruling** : Does not mention business use as justification for absorption.
   - **Court of Appeal Ruling** : It emphasizes that corporate practice can derogate from the absorption principle, subject to a favorable reiteration by the employer, and states that such practice has not been demonstrated. *Conclusion:* The two rulings address the same legal dispute, but reach different conclusions for the various appellants. While the court of first instance upholds the workers' claims, the Court of Appeal upholds the first instance decision for Gini and Mori, but excludes Voni from the dispute. The differences in the outcomes and litigation costs highlight the divergences in legal interpretations and applications of legal principles regarding wage absorption.

Response generated with: **DQL**

› View details

A few more details …

Extra slides for QA

Code is open source: https://github.com/unimib-datAI/VCL
If you are interested in a demo, get in touch: matteo.palmonari@unimib.it

BACKUP

Use case & corpus — the judge's task

The corpus — one real court case

- First-instance Judgment (Judgment_1°)
- Appeal
- Second-instance ruling / Defence memory brief
- Plus cited rulings (e.g. a Cassation judgment)

The judge's task

Draft a report (relazione) that extracts and organizes information across the documents — material the judge then refines while writing the opinion.

Why it is hard

- Documents heavily cross-reference one another
- The same concept (e.g. “decision”) appears in several documents
- Some cited rulings carry no own text — flagged as “conf.”
- Answers must stay faithful, scoped and traceable to the source
- Generic chunk retrieval blurs document boundaries

VCL-Q IN ACTION

From a verbal request to a controllable program

> *“ Do the failures described by customers violate the thermal-safety specs in the manual — and did the test report see this coming? ”*

```
#1 = SEMANTIC EXTRACT [Complaint_A, Complaint_B], [Failure modes]
#2 = SEMANTIC EXTRACT [Technical_Manual], [Thermal safety specs]
#3 = COMPARE [#1, #2], "highlight any violations"
#4 = COMPARE [#3, Test_Report], "was this already observed?"
```

Every step is explicit, scoped to named documents, and inspectable before it runs.

RAG
Finds the most relevant text for the whole question at once, and lets the model figure out the rest.

VCL
Control exactly what to look for, where, and how, so you can **inspect, customize and trust** every step.

Same language works across legal, industrial and clinical documents.

USER-DEFINED ONTOLOGY (VCL-D)

VCL-Definitions: teaching VCL the language of your domain

VCL-D defines how **user-specified elements** are interpreted and constrained — manually crafted prior to deployment, reused across queries

VCL-D_source

characterizes each source document

`label` the canonical document name used in queries
`synonyms` alternative phrasings the user might say
`description` what the document is and typically contains

e.g. a court ruling

```
label        Judgment_1°
synonyms     "first-instance ruling"
description  "the ruling issued by the trial court"
```

VCL-D_what

defines each target element

name the label of a category of text to look for inside documents
semantic description definition of the category
availability which sources may contain it

e.g., a decision:

```
name          decision
description   "the operative part of a ruling …"
availability  [Judgment_1°, Judgment_2°]
```

Why user-defined? adapts to any domain *without modifying the core engine* · knowledge base **explicit · auditable · controllable**

Preprocessing & Translation (NL → VCL)

Preprocessor

MACRO-PLANNING

An LLM breaks the request into sub-tasks.

“Create a table comparing the decisions DOC1 and DOC2.
Then do the same for the litigation costs.”

```
T1: Compare the decisions in DOC1 and DOC2 using a
table
T2: Compare the litigation costs in DOC1 and DOC2
using a table
```

Translator

NL → VCL

Each task becomes one VCL clause, filled by specialized agents grounded in the ontology:

Command Classifier	⟨command⟩
Sources Extractor	⟨from⟩
What Extractor	⟨what⟩
Conditions Extractor	⟨conditions⟩

```
COMPARE [DOC1, DOC2 ["decisions"] "using a table"
COMPARE [DOC1, DOC2 ["costs"] "using a table"
```

Planning & execution (VCL → answer)

Planner

MICRO-PLANNING · DETERMINISTIC

Compares the VCL request with ontology instances and decomposes it on ⟨from⟩ and ⟨what⟩ into atomic operations.

```
T1.1: EXTRACT [DOC 1] ["decisions"]
T1.2: EXTRACT [DOC 2] ["decisions"]
T1.3: COMPARE [T1.1, T1.2] "using a table"

T2.1: SEARCH [DOC 1] ["costs"]
T2.2: SEARCH [DOC 2] ["costs"]
T2.3: COMPARE [T1, T2] ["decisions"] "using a table"

INTEGRATE [T1.3, T2.3] “merging tables”
```

Executor

RETRIEVE + RUN

Document retrieval — restricted to the documents named in the request.

One LLM-based tool per command. Each tool prompt is enriched with:

- context
- additional conditions
- ontology element definitions

`SEARCH tool` `SEM. EXTRACT tool`

`COMPARE tool` `SUMMARIZE tool`