MINISTRY OF HIGHER EDUCATION,
SCIENTIFIC RESEARCH AND INNOVATION
SULTAN MOULAY SLIMANE UNIVERSITY
NATIONAL SCHOOL OF APPLIED SCIENCES OF KHOURIBGA
Department of Mathematics and Computer Science

PROJECT REPORT

Evidence-Grounded Call Intelligence: Structured Extraction and Agentic Conversational Question Answering

A Structured Extraction Contract and Text-to-Query Agent for Business Call Analysis

Presented by

EL ABBASSI Mouad & ZAHID Achraf

Carried out at

Supervised by

LAMGHARI Nidal	Academic Supervisor	ENSA Khouribga
OUBAIBOU Adil	Professional Supervisor	Leyton CognitX
AIT OUALLANE Abderrahmane	Professional Supervisor	Leyton CognitX

Jury Members

BOUCHANTOUF Aya	Jury President	ENSA Khouribga
SARSARI Kaoutar	Examiner	ENSA Khouribga

Academic Year

2025–2026

Acknowledgements

At the completion of this project, we would like to express our sincere gratitude to all those who contributed, directly or indirectly, to the successful realization of this work.

First, we would like to warmly thank our company supervisor, **Mr. Adil OUBAIBOU & Abderrahmane AIT OUALLANE**, for his guidance, availability, valuable advice and trust throughout the project. His feedback helped us continuously refine our technical choices, confront our approach with real project requirements and improve the overall quality of our work.

We would also like to thank our academic supervisor, **Mme LAMGHARI Nidal & Mr AZOUANI Abderrahim**, for the academic guidance, recommendations and support provided throughout the project. The feedback received helped us take a broader perspective on our work and structure our methodology with greater rigor.

We extend our sincere thanks to the entire **DATALA** team, particularly the **Applied ML** and **Generative AI** teams, for their welcome, availability and the professional environment provided during this project. The discussions with the different team members allowed us to better understand the technical and business challenges related to extracting and exploiting information from DigiVoice calls, while significantly enriching our professional experience.

We are also grateful to the **National School of Applied Sciences of Khouribga** and **Sultan Moulay Slimane University** for the education, technical knowledge and academic environment provided throughout our studies, which prepared us to undertake this project.

Finally, we would like to thank everyone who supported and encouraged us during the completion of this work. Their assistance, discussions and encouragement were valuable contributions to the successful completion of this project.

Abstract

DigiVoice calls contain valuable business information, but this information is distributed across long and unstructured transcripts, making direct retrieval incomplete and difficult to trace.

In this project, we designed a structured extraction pipeline that transforms timestamped call transcripts into a typed `Summary` with evidence references, deterministic metadata and a separate semantic representation for retrieval. The approach was shaped by the manual analysis of 30 real calls and by several benchmarks covering prompt versions, models and chunking strategies.

The retained pipeline uses adaptive chunking, independent extraction and controlled merging. The evaluation methodology was also redesigned to compare Legacy and Improved extraction fairly using prompt aware references and a common information ledger. On the two evaluated calls, the Improved version increased common information capture by an equal weight average of 14.2 percentage points.

On top of this structured information layer, we designed a question answering agent combining intent classification, planning, retrieval, conversational state and grounded answer generation. The resulting architecture provides a more structured, traceable and evaluable way to exploit DigiVoice calls.

Contents

Acknowledgements i

Abstract ii

1 Project Context and Problem Statement 1

1.1 Project scope . . . 1
1.2 The central problem . . . 1
1.3 Why direct transcript search was not enough . . . 2
1.4 Engineering objectives . . . 2

2 State of the Art and Evolution of Information Extraction 4

2.1 Introduction . . . 4
2.2 Classical Information Extraction . . . 5
2.2.1 Rule-Based and Schema-Specific Extraction . . . 5
2.2.2 Statistical Sequence Labelling . . . 6
2.3 Contextual Representation and Transformer-Based Extraction . . . 6
2.3.1 The Transformer Transition . . . 6
2.3.2 Span and Joint Extraction Models . . . 6
2.4 From Classification to Generative Information Extraction . . . 7
2.4.1 Structured Prediction as Generation . . . 7
2.4.2 Instruction-Tuned Information Extraction . . . 8
2.5 Conversation and Meeting Understanding . . . 8
2.5.1 Why Conversation Is Different from Written Documents . . . 8
2.5.2 Summarization Versus Structured Extraction . . . 9
2.6 The Long-Context Problem . . . 9
2.6.1 Context Length Is Not Equivalent to Context Utilization . . . 9
2.6.2 Implications for Segmentation . . . 10
2.7 Hierarchical Processing and Multi-Stage Aggregation . . . 10
2.8 From Extraction to Retrieval-Augmented Question Answering . . . 11
2.8.1 Semantic Retrieval . . . 11
2.8.2 Retrieval-Augmented Generation . . . 11
2.9 Graph-Based Retrieval and Global Questions . . . 11
2.10 Agentic Retrieval and Tool-Oriented Question Answering . . . 12

2.11 Structured Output Validation as an Engineering Requirement 12
2.12 Evidence, Grounding and Evaluation 13
2.12.1 Why Structural Validity Is Not Enough 13
2.12.2 Evaluation Beyond One Aggregate Score 14
2.13 Evolution of the Design Space 14
2.14 Positioning of the DigiVoice Approach 15
2.15 State-of-the-Art Synthesis 16

3 Real Call Audit and Data Understanding 18
3.1 Why we audited calls before finalizing a schema 18
3.2 Observed call purposes 18
3.3 Recurring business interaction pattern 19
3.4 Source reliability findings 19

4 Business Questions and the Extraction Contract 22
4.1 From questions to information requirements 22
4.2 Contract terminology 23
4.3 The earlier contract 23
4.4 The current Summary contract 23

5 Prompt Engineering and Contract Evolution 26
5.1 Why prompt engineering became a core design problem 26
5.2 Prompt milestones 27
5.3 V7.1 design principle 27
5.4 Evidence contract 28
5.5 Participant admission rule 28
5.6 Semantic admission gates 28
5.7 Project eligibility discipline 29
5.8 Chunk locality and boundary discipline 29
5.9 Merge prompt discipline 30
5.10 Separate semantic summary 30

6 Long Call Processing and Adaptive Chunking 31
6.1 Why segmentation became necessary 31
6.2 Whole call extraction as the baseline 32
6.3 Fixed chunk experiments 32
6.4 Retained adaptive heuristic 33

7 Model, Prompt and Processing Benchmarks 35
7.1 Initial formal benchmark . . . 35
7.2 Why the benchmark did not end the investigation . . . 36
7.3 Luna and Terra long call comparison . . . 37
7.4 Short call V3 and V4 checks . . . 37

8 Extraction Strategy and Controlled Merge 39
8.1 Strategies that we tested . . . 39
8.2 Why independent extraction was retained . . . 39
8.3 Merge invariants . . . 40
8.4 Model pairing for extraction and merge . . . 41

9 Post Processing, Metadata and Elasticsearch Representation 42
9.1 Why the model does not own every output field . . . 42
9.2 Three information layers used for querying . . . 42
9.3 Derived metadata as a cost control mechanism . . . 43
9.4 Semantic summary as a retrieval representation . . . 43

10 Evaluation Methodology and Ground Truth 44
10.1 Why the original evaluation approach was not enough . . . 44
10.2 Three evaluation phases . . . 44
10.3 Soft and strict matching . . . 45
10.4 Native contract scores . . . 46
10.5 Common information soft capture . . . 46
10.6 Category gains and regressions . . . 47
10.7 Gold reference reproducibility package . . . 48

11 Retained Extraction Pipeline and Cost Decision 49
11.1 Current processing flow . . . 49
11.2 Reported batch cost . . . 50
11.3 Decision statement . . . 50

12 From Structured Calls to a Question Answering Agent 51
12.1 Why an agent layer was necessary . . . 51
12.2 Architecture shown in the project presentation . . . 51
12.3 Earlier end to end baseline . . . 53
12.4 Why the agent architecture evolved again . . . 53

13 Architecture 54

13.1 Intent Classification 54
13.1.1 Current four field contract 54
13.1.2 Intent categories and answer shapes 54
13.1.3 Follow up resolution 54
13.1.4 Interface issues discovered during review 55
13.2 Router, Planning and Execution 55
13.2.1 Router responsibility 55
13.2.2 Hybrid Router design 55
13.2.3 Why planning uses business operations rather than raw DSL . . . 56
13.2.4 Representative planning cases 56
13.2.5 Data model issues that affect routing 57
13.3 State, Memory and Conversational Continuity 58
13.3.1 Why state exists 58
13.3.2 Earlier state model 58
13.3.3 Current design direction 58
13.4 Answer Composition and Grounding 59
13.4.1 Answer Composer responsibility 59
13.4.2 Why the final answer layer is separate 59
13.4.3 Answer shape 59

14 Agent Validation, Demo and Maturity 60
14.1 What has been implemented 60
14.2 What has been validated in the latest Router design 60
14.3 What we should not claim yet 60
14.4 Demo limitations observed 61

15 Challenges, Failures and Engineering Lessons 62
15.1 Identity and diarization 62
15.2 Long context versus segmentation 62
15.3 Cumulative state regression 62
15.4 Schema growth versus semantic precision 62
15.5 Evaluation fairness 63
15.6 Metrics can disagree 63
15.7 Model API compatibility 63
15.8 Routing and data semantics 63

16 Roadmap and Remaining Validation 64

16.1 Router integration benchmark 64
16.2 Retrieval improvements 64
16.3 Production observability 64

Bibliography **66**

A Foundation Question Set **71**
A.1 Single call comprehension 71
A.2 Single call quality 71
A.3 Consultant performance 71
A.4 Time and volume 71
A.5 Content themes and client history 72

B Current Prompt Contract in Report Form **73**
B.1 Global chunk rules 73
B.2 Evidence rules 73
B.3 Participant rules 74
B.4 Project eligibility rules 74
B.5 Boundary rules 74
B.6 Merge rules 75
B.7 Semantic summary rule 75

C Benchmark Interpretation Rules **76**
C.1 Do not mix scorecards 76
C.2 Do not mix cost scopes 76
C.3 Distinguish model choice from contract choice 76

List of Tables

2.1 Evolution of information extraction approaches 15

3.1 Primary purposes in the 30 audited calls 18
3.2 Audit findings and extraction implications 20

4.1 Representative business questions and required information 22
4.2 Earlier contract representation 23
4.3 Current Summary collections and semantic role 24

5.1 Main prompt and schema milestones 27
5.2 Examples of current semantic admission gates 29

6.1 Recorded input token profile 31
6.2 Long call Luna V3 chunk sweep 33
6.3 Adaptive target chunk count 33

7.1 Complete 12 configuration initial benchmark 35
7.2 Long call Luna and Terra comparison under V3 and 10k 37
7.3 Recorded short call comparisons 38

8.1 Processing strategies and retained decision 39

9.1 Logical call information layers 42

10.1 Native prompt aware contract scores 46
10.2 Common information soft capture 46
10.3 Selected category changes in the two common ledgers 47

11.1 Reported batch processing cost 50

12.1 Architecture components and the question each component answers . . . 52

13.1 Representative Router challenges 57

14.1 Current evidence maturity by subsystem 61

List of Figures

1.1 Project overview showing the path from raw DigiVoice call to structured extraction, indexed call knowledge, agent planning, retrieval and grounded answer. 2

2.1 Evolution of information extraction from rule-based systems to evidence-grounded LLM pipelines 5
2.2 Positioning of the DigiVoice approach within the evolution of information extraction 16

3.1 Horizontal bar chart of the seven audited call purposes using black, dark gray and light gray only. 19
3.2 A two column cause and effect diagram connecting the main real call audit findings to the extraction rules they motivated. 21

4.1 A monochrome schema map showing the 16 Summary collections grouped into people, organizations, projects, needs and friction, decisions and follow up, documents and history, reusable facts and unresolved information . . 25

5.1 Prompt architecture diagram showing field contract, chunk task, merge task, timestamp evidence and separate semantic summary generation. . . 30

6.1 Range plot showing minimum, median, average and maximum input sizes for cleaned JSON and JSON plus prompt. 32
6.2 Line chart or horizontal dot plot of the five Luna V3 chunk sweep quality values. 33

7.1 Scatter plot of all 12 configurations with latency on the horizontal axis and audited quality on the vertical axis, with labels for every configuration. . 36
7.2 Four panel diagnostic showing projects 4 of 4, participants 8 of 8 with 3 extra, decisions 0 of 6 and terms 0 of 8. 37

8.1 Two lane comparison: cumulative shared state versus independent chunk extraction followed by controlled merge. Emphasize the retained independent path 40

10.1 Three phase evaluation diagram with Legacy native evaluation, Improved native evaluation and shared common information comparison. 45

10.2 Paired horizontal bars for call 6830 and call 7517 showing Legacy versus Improved common information capture on the same 0 to 100% scale. . . 47
10.3 Dot comparison chart grouped into progress and regressions for projects, problems, rules and facts, people, decisions and historical events. 48

11.1 End to end extraction flow: source call, adaptive chunks, Luna extraction, Luna merge, validated Summary, application processing, Elasticsearch document and separate semantic summary request. 49
11.2 Two horizontal bars for reported cost per 50 calls: Legacy 0.87 USD and Improved 1.00 USD, with a small annotation for the 0.13 USD difference. 50

12.1 Monochrome version of the presentation architecture: User, Intent Classifier, Planner and Router, execution tools, call database, state management, Answer Composer and Final Answer. 52

13.1 Latest Router design from four field QueryIntent through confidence gate, semantic planner, entity and date resolution, capability compiler, plan validator, Executor and grounded Answer Generator. 56
13.2 Conversation state diagram showing previous question and answer entering the classifier, explicit query scope entering the Router, and the final answer being written back to session memory 59

Chapter 1

Project Context and Problem Statement

1.1 Project scope

The project focuses exclusively on the DigiVoice call source. We work from transcribed calls and their associated source metadata and evaluation information. Our objective is to make this source usable for structured search, filtering, analysis and natural language question answering.

The project is therefore broader than a summarization feature but narrower than a generic enterprise knowledge system. The scope is one call domain with a controlled ingestion contract and a dedicated conversational querying layer.

1.2 The central problem

> How can we transform long, noisy and timestamped DigiVoice conversations into structured, searchable and traceable business knowledge, then answer natural language questions over that knowledge while controlling omissions, unsupported claims, latency and cost?

The business value inside a call is not stored as a clean table. It is expressed through natural conversation. Important facts are distributed, repeated, corrected and sometimes implied through context. Raw transcripts create five practical difficulties.

1. Information is distributed over many turns and may only become interpretable after later context.
2. Generic speaker labels do not safely identify named people.
3. Similar concepts such as a need, a pain point, an objection, a technical challenge and an uncertainty can be confused.
4. Long calls create model input, latency and consistency problems.
5. A question answering agent needs structured filters, semantic content and source evidence, not only a paragraph summary.

1.3 Why direct transcript search was not enough

A pure transcript retrieval approach can find relevant text, but it does not automatically provide reliable aggregation, entity state, action status or cross call comparison. For example, the question "Which calls had objections but no next step?" requires the system to reason about the existence of one type of business object and the absence of another. The question "Top 5 SDRs by average score this quarter" requires date normalization, a numeric metric, grouping and sorting. The question "What are Acme's pain points?" requires entity resolution and retrieval of a specific semantic collection across possibly several calls.

We therefore separated the project into two connected layers. The first layer converts calls into a stable information model. The second layer converts a user question into a grounded retrieval and answer process.

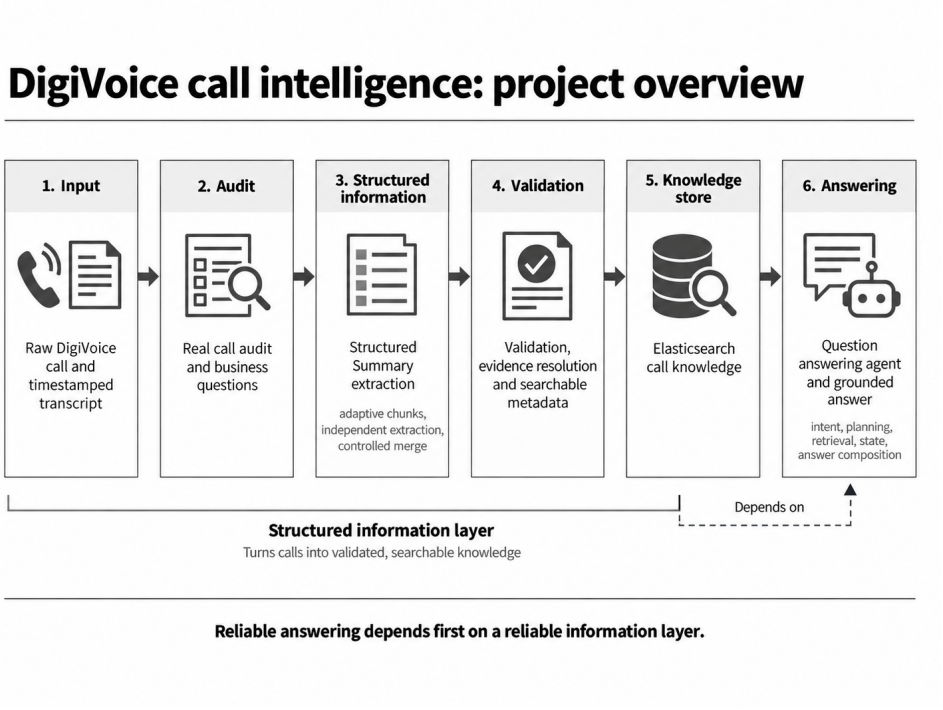


Figure 1.1: Project overview showing the path from raw DigiVoice call to structured extraction, indexed call knowledge, agent planning, retrieval and grounded answer.

1.4 Engineering objectives

We defined six objectives for the project.

1. Preserve the business meaning that is repeatedly required by real questions.
2. Preserve provenance so important claims can be traced back to the call.
3. Minimize unsupported additions and identity assumptions.

4. Handle long and heterogeneous call lengths with predictable processing.
5. Make the result efficient to query in Elasticsearch.
6. Build an agent whose decisions are inspectable and whose final answer is based on retrieved evidence.

These objectives connect data science, natural language processing, structured output validation, ETL design, Elasticsearch retrieval, prompt engineering and experimental evaluation. The general Leyton data science checklist also emphasizes business focused questions, ETL, Elasticsearch, experiment design, precision, recall and F1 as core engineering practices; in this project those practices were applied to the DigiVoice use case rather than treated as theory alone.

Chapter 2

State of the Art and Evolution of Information Extraction

2.1 Introduction

Information extraction aims to transform unstructured natural language into structured representations that can be searched, analyzed and reused by downstream systems. Over time, the field has evolved from manually engineered rules and domain-specific extraction systems to statistical sequence models, deep neural architectures, pretrained Transformers and, more recently, generative Large Language Models capable of producing complex structured outputs from natural-language instructions [2, 15, 16].

This evolution is particularly relevant to business conversations. A long customer or sales call does not usually express all useful information in one sentence or one contiguous segment. Participants may be introduced at the beginning of a conversation, technical constraints can appear later, a project may be refined gradually, an objection can be raised and subsequently resolved, and a decision may only become explicit at the end of the call. Consequently, extracting reusable information from a conversation requires more than classical named entity recognition or free-form summarization.

In the context of this project, we were interested in a specific question: how can a long timestamped DigiVoice transcript be transformed into a structured, evidence-grounded representation that can subsequently support reliable retrieval and question answering?

To understand the design space, we studied several generations of information extraction systems. This chapter presents that evolution and identifies the ideas that influenced the architecture developed in this project. It is important to distinguish between approaches reviewed as related work and components actually implemented. In particular, RAPTOR and GraphRAG are included because they illustrate important hierarchical and graph-based approaches to long-context information organization, but they are not claimed as implemented components of the retained DigiVoice extraction pipeline.

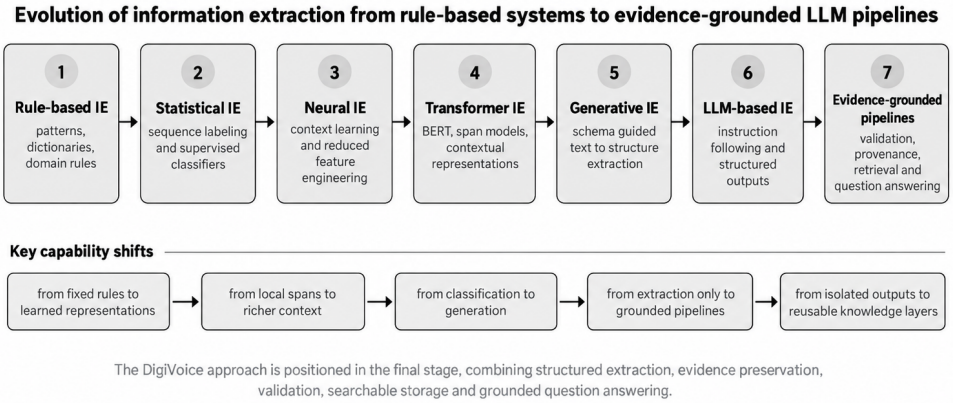


Figure 2.1: Evolution of information extraction from rule-based systems to evidence-grounded LLM pipelines

2.2 Classical Information Extraction

2.2.1 Rule-Based and Schema-Specific Extraction

Early information extraction systems were generally designed around a predefined domain and a limited set of target entities or relations. A system could, for example, be configured to identify people, organizations, dates or specific relations using lexical rules, regular expressions, dictionaries, syntactic patterns and manually maintained domain knowledge.

The principal advantage of these approaches was interpretability. A rule could often be traced directly to the extracted result. However, the engineering cost increased significantly when the domain, vocabulary or required relation changed. The dependence on manually constructed extraction logic therefore limited scalability and cross-domain reuse.

The introduction of Open Information Extraction represented an important change in this paradigm. Banko et al. proposed OpenIE as a mechanism for extracting relational tuples without requiring every relation type to be specified in advance [1]. Instead of designing a separate extractor for every relation, the system attempted to discover a broad set of relational facts directly from text.

This was an important conceptual step toward more general information extraction. Nevertheless, early OpenIE systems still depended heavily on linguistic preprocessing and extraction patterns. Their output was also typically based on relational tuples rather than rich business objects containing states, provenance, evidence and domain-specific semantics.

2.2.2 Statistical Sequence Labelling

The next generation increasingly formulated extraction as supervised sequence labelling. Tasks such as Named Entity Recognition became token classification problems in which models learned labels from annotated examples rather than depending entirely on manually specified rules.

This family of methods improved portability and reduced manual rule engineering, but required labelled training corpora and remained strongly tied to the predefined label inventory. It was particularly effective for local extraction tasks such as identifying people, organizations, locations and other span-level entities.

The development of neural sequence models further reduced the need for handcrafted features. Lample et al. demonstrated that neural architectures based on recurrent networks and character-level representations could obtain strong NER performance without relying on extensive language-specific feature engineering [3].

OpenIE also evolved in this direction. Stanovsky et al. formulated supervised OpenIE as sequence tagging [4], while Cui et al. explored an encoder-decoder neural architecture for OpenIE [5]. These works illustrate the transition from explicit linguistic patterns toward learned representations.

2.3 Contextual Representation and Transformer-Based Extraction

2.3.1 The Transformer Transition

A major change in natural language processing followed the introduction of the Transformer architecture by Vaswani et al. [36]. Self-attention allowed contextual relationships between tokens to be represented without relying exclusively on sequential recurrent processing.

BERT subsequently demonstrated that deeply bidirectional Transformer representations could be pretrained on large unlabelled corpora and then fine-tuned for downstream language understanding tasks [7]. This substantially changed the information extraction landscape. Instead of learning representations only from a relatively small task-specific dataset, extraction systems could build on general linguistic representations learned during large-scale pretraining.

For information extraction, this was especially useful because identifying a business entity, relation or event often depends on context rather than the surface form of a word.

2.3.2 Span and Joint Extraction Models

Several approaches moved beyond independent token classification and represented candidate spans or interactions between extracted objects.

SpanBERT explicitly modified Transformer pretraining to improve span representations and demonstrated benefits on relation extraction, coreference and question answering tasks [8].

DyGIE++ proposed a multi-task framework for entities, relations and events using contextualized span representations and graph propagation [9]. The approach demonstrated that extraction should not always be viewed as independent local decisions because entities and relations can depend on wider document context.

Zhong and Chen later showed that a relatively simple pipelined design could perform strongly for joint entity and relation extraction when separate representations were learned for the two subtasks [10].

These systems introduced two ideas that remain relevant to our project.

First, extraction quality depends on contextual representation rather than simple keyword matching. Second, separating responsibilities between stages can sometimes be preferable to asking one model to perform every task jointly.

2.4 From Classification to Generative Information Extraction

2.4.1 Structured Prediction as Generation

The emergence of powerful sequence-to-sequence models allowed information extraction to be reformulated as generation.

TANL, proposed by Paolini et al., represented structured prediction problems as translation between augmented natural-language representations [11]. Instead of introducing a dedicated classifier for every output component, the system generated an augmented sequence containing the desired structured information.

This direction became more explicit with Unified Information Extraction. Lu et al. proposed UIE, a text-to-structure framework capable of representing multiple information extraction tasks through a common structured extraction language and schema-based prompting mechanism [12]. The same framework could address entities, relations, events and sentiment extraction.

This idea is highly relevant to our work because the DigiVoice extraction problem is also heterogeneous. A useful call representation must simultaneously capture people, companies, projects, needs, pain points, objections, decisions, actions, documents, historical events, open questions and other business objects. Building a completely separate model for each information type would result in a complex and difficult-to-maintain pipeline.

2.4.2 Instruction-Tuned Information Extraction

Large Language Models extended the generative extraction paradigm further. Instead of requiring a specialised architecture for every schema, a general language model could receive an instruction describing both the extraction task and the expected structure.

InstructUIE explored multi-task instruction tuning for unified information extraction and introduced a large collection of IE tasks represented through instructions [13]. This work demonstrated the growing importance of explicit task descriptions and schemas when using generative models.

GPT-RE investigated in-context learning for relation extraction using large language models [14]. Its results also demonstrate that strong general-purpose language models do not automatically solve information extraction. Demonstration selection, task representation and label semantics remain important.

Recent surveys describe this transition as a broader move from discriminative information extraction toward generative information extraction [15, 16]. The model is no longer restricted to selecting a class for each token. It can instead generate structured objects whose format is controlled by a schema or instruction.

For our project, this paradigm enables one extraction request to populate a rich typed `Summary`. However, the increased flexibility also creates new risks. A generative model can create unsupported information, combine semantically distinct objects, omit required fields or produce relationships that are linguistically plausible but not grounded in the source.

Research on LLM-based information extraction has shown that these systems can form spurious associations between labels and extracted content [17]. Consequently, flexible generation does not remove the need for strict admission criteria, evidence and post-generation validation.

2.5 Conversation and Meeting Understanding

2.5.1 Why Conversation Is Different from Written Documents

Business calls introduce difficulties that do not appear to the same degree in conventional document extraction.

Conversation is sequential, interactive and often redundant. Speakers may interrupt one another, refer to previously introduced entities using pronouns, revise earlier statements or answer questions indirectly. Important information can therefore depend on conversation history.

Dialogue summarization datasets such as DialogSum were introduced to study real-life conversational summarization and highlight challenges including discourse structure, coreference, ellipsis, pragmatics and conversational context [18].

Meeting summarization presents an even closer analogy to business calls. QMSum introduced query-based meeting summarization and showed that a single generic summary is not necessarily sufficient for long multi-topic conversations [19]. A user may require a summary of decisions, another may need action items, and another may need information about one specific technical subject.

A survey by Rennard et al. similarly identifies meeting summarization as a distinct problem due to the length, multi-party nature and conversational structure of meetings [20].

2.5.2 Summarization Versus Structured Extraction

These observations explain why we distinguish semantic summarization from structured extraction.

A fluent abstractive summary is useful for reading and semantic retrieval, but it is not necessarily an appropriate source of truth for downstream business queries. Summarization is intentionally compressive. A model may decide that a specific person name, minor decision, objection or historical detail is not important enough to include.

Structured extraction has a different objective. If a business object satisfies its admission rules, it should be represented even if it would not appear in a short narrative summary.

This distinction directly motivated the separation in our architecture between the typed `Summary`, which is treated as the structured information layer, and the semantic summary, which is generated separately as a retrieval-oriented representation.

2.6 The Long-Context Problem

2.6.1 Context Length Is Not Equivalent to Context Utilization

Long business calls can contain tens of thousands of tokens. A natural solution would appear to be sending the entire transcript to a sufficiently large-context language model.

However, two separate issues must be considered.

The first is computational. Standard Transformer self-attention has a quadratic relationship with sequence length [36]. Architectures such as Longformer introduced local and global attention patterns to support longer documents more efficiently [21]. BigBird similarly proposed sparse attention to reduce the computational cost of processing long sequences [22].

The second issue is semantic utilization. A model accepting a large input window does not necessarily use every part of that window equally well. Liu et al. demonstrated this explicitly in *Lost in the Middle*, showing that performance can deteriorate when relevant information is located in the middle of a long context [41].

This result is particularly important for call intelligence. Relevant information may appear anywhere in a transcript, and its position is not under the control of the extraction system.

Therefore, maximum context size should not automatically be interpreted as the optimal processing size.

2.6.2 Implications for Segmentation

Chunking reduces the amount of information presented to an extraction model at one time, but introduces a different challenge: information belonging to the same business object can be distributed across multiple chunks.

Small chunks increase local focus but also increase the number of model requests and the complexity of reconciliation. Large chunks preserve more context but increase token cost, latency and the possibility that information is overlooked.

Consequently, chunk size should be treated as an empirical system parameter, not as a fixed theoretical constant.

This reasoning is consistent with our experimental approach. Instead of selecting an arbitrary chunk size, we benchmarked several segmentation configurations and later adopted an adaptive strategy based on the observed input lengths.

2.7 Hierarchical Processing and Multi-Stage Aggregation

A growing body of research avoids treating long documents as one flat sequence.

RAPTOR constructs a hierarchical representation by recursively embedding, clustering and summarizing text chunks. Retrieval can subsequently occur at different levels of abstraction [42]. The main idea is that information needed for a complex question may exist at several granularities, and a hierarchical representation can preserve both local details and broader document structure.

RAPTOR is a retrieval architecture rather than the extraction mechanism used in this project, but it illustrates an important principle: long inputs can be processed locally and subsequently reconciled into a higher-level representation.

Our retained extraction architecture applies a related principle to structured objects. Each transcript chunk is extracted independently. The resulting local summaries are then provided to a controlled merge stage.

The distinction is important. The merge operation is not asked to rediscover the original transcript. Instead, it receives already extracted structured objects and is responsible for reconciliation, deduplication, attribute preservation and conservative identity resolution.

This reduces the risk associated with cumulative mutable state, in which each new

chunk can inadvertently overwrite information extracted from previous chunks.

2.8 From Extraction to Retrieval-Augmented Question Answering

2.8.1 Semantic Retrieval

Once structured information has been extracted, the next problem is locating the relevant call or objects for a user question.

Semantic representation learning significantly changed information retrieval. Sentence-BERT demonstrated that dense sentence embeddings can support efficient semantic similarity search [39]. Dense Passage Retrieval subsequently demonstrated the effectiveness of learned dense representations for open-domain question answering [38].

These methods complement traditional lexical retrieval. Lexical approaches are particularly effective when users provide exact names or terms, while semantic representations are useful when the wording of a question differs from the wording stored in the source.

2.8.2 Retrieval-Augmented Generation

Lewis et al. introduced Retrieval-Augmented Generation as a framework combining a generative model with an external non-parametric knowledge source [37]. Instead of relying only on knowledge stored in model parameters, the system retrieves relevant external information and conditions generation on that evidence.

This architecture is conceptually aligned with the question-answering portion of our project. The language model should not invent what occurred in a business call from general world knowledge. The answer must be grounded in retrieved DigiVoice information.

However, conventional RAG often retrieves unstructured text chunks. Our architecture introduces an intermediate structured information layer. Retrieval can therefore operate over fields such as clients, participants, projects, pain points, decisions and actions, in addition to semantic text.

This distinction is useful for business questions that require deterministic filters or aggregations rather than only semantic similarity.

2.9 Graph-Based Retrieval and Global Questions

Standard RAG is often effective for local questions where the answer can be found in a small number of passages. More global questions are more difficult.

GraphRAG was proposed to address query-focused reasoning over large private corpora by constructing an entity graph, organizing the graph into communities and generating

summaries at multiple levels [44]. The GraphRAG documentation describes an indexing pipeline that extracts entities, relationships and claims before creating community-level representations [35].

Graph-based organization is relevant to call intelligence because conversations naturally contain relationships between people, companies, projects, documents, actions and decisions. Such representations could eventually support questions that require relationships across many calls.

Nevertheless, the current DigiVoice implementation described in this report should not be confused with GraphRAG. Our retained production-oriented representation is a structured Elasticsearch call document rather than a GraphRAG community hierarchy.

GraphRAG is therefore presented as an important related direction and a possible future extension rather than as an implemented project component.

2.10 Agentic Retrieval and Tool-Oriented Question Answering

The evolution from retrieval systems to language-model agents introduces another architectural shift. Instead of mapping every user question to the same retrieval operation, an agent can determine which sequence of actions is required.

ReAct demonstrated the value of combining reasoning with actions that interact with external information sources [46]. This separation between reasoning and tool interaction is relevant to systems in which some questions require simple retrieval while others require filtering, aggregation or multiple dependent steps.

Self-RAG further explored adaptive retrieval and self-reflection instead of retrieving a fixed amount of information for every request [47].

These works influenced how we reason about the question-answering layer. The Intent Classifier should understand and reformulate the question but should not directly construct an Elasticsearch query. Planning and capability selection belong to the Router, while actual access to Elasticsearch belongs to an execution layer.

This separation reduces the responsibilities assigned to each LLM call and makes the resulting system easier to validate.

2.11 Structured Output Validation as an Engineering Requirement

Generative information extraction produces flexible outputs, but production systems require predictable contracts.

In our implementation, Pydantic models define the expected structure of the extracted

information. Pydantic validates data against typed model fields and can generate JSON Schema representations from those models [32].

This does not guarantee that the semantic content of an extracted object is correct. A syntactically valid pain point can still be unsupported by the conversation. Nevertheless, structural validation provides an important boundary between probabilistic generation and deterministic application code.

The same principle applies to the API layer. FastAPI can use typed Pydantic response models for validation, serialization and OpenAPI schema generation [33].

Elasticsearch then provides the persistent searchable layer. Its structured fields and aggregation framework make it possible to support both document retrieval and analytical operations over indexed calls [34].

The resulting architecture therefore combines probabilistic and deterministic components:

LLM extraction → typed validation → deterministic post processing → searchable storage.

2.12 Evidence, Grounding and Evaluation

2.12.1 Why Structural Validity Is Not Enough

A central challenge with LLM-based extraction is that a plausible output may still be unsupported.

This is especially problematic in business intelligence because the extracted representation can later be treated as a factual database. An invented participant, incorrect project association or incorrectly inferred decision can propagate from extraction to retrieval and eventually appear in a final answer.

The literature on LLM-based information extraction continues to document failure modes associated with generative extraction [15, 17]. For this reason, we consider source evidence to be part of the extracted object rather than optional explanatory metadata.

In the DigiVoice pipeline, transcript timestamps provide a mechanism for connecting an extracted object back to its source. Post processing can later resolve that evidence into readable text and speaker information.

This architecture creates three different levels of verification:

1. the model must produce an object admitted by the extraction rules;
2. the object must satisfy the typed structural contract;
3. the object must retain evidence that can be inspected against the original call.

2.12.2 Evaluation Beyond One Aggregate Score

Evaluation must similarly operate at more than one level.

Traditional IE evaluation commonly uses precision, recall and F1 against an annotated reference. For structured LLM extraction, however, the definition of an extraction unit becomes more complex. A model can identify the correct business object but omit one attribute, associate the correct fact with the wrong project or express the same information using a semantically equivalent form.

For retrieval-augmented systems, frameworks such as RAGAS distinguish retrieval quality from answer faithfulness and generation quality [31]. The broader lesson is that a single end-to-end metric can hide the location of a failure.

We applied the same principle to extraction. Our evaluation distinguishes field-level coverage, strict and soft matching, evidence quality and schema-specific behaviour. When comparing two different schemas, we also separate native contract evaluation from the common-information comparison.

This is necessary because a richer schema may contain fewer or more evaluation units than an earlier schema. Directly comparing two native F1 scores with different denominators can therefore produce a misleading conclusion.

2.13 Evolution of the Design Space

Table 2.1 summarizes the principal evolution of information extraction approaches and their relevance to long business conversations.

Table 2.1: Evolution of information extraction approaches

Generation	Typical approach	Principal strength	Principal limitation for business calls
Rule based	Patterns, dictionaries, linguistic rules	High interpretability and deterministic behaviour	Expensive domain adaptation and poor scalability to many business concepts
Statistical IE	Sequence labelling and supervised classifiers	Learns extraction behaviour from annotated data	Requires labelled data and predefined labels
Neural IE	BiLSTM, neural OpenIE, span models	Reduces manual feature engineering and captures context	Often task specific and dependent on supervised datasets
Transformer IE	BERT, SpanBERT, contextual span extraction	Strong contextual representations and transfer learning	Long documents remain difficult and output schemas remain task dependent
Generative IE	TANL, UIE and schema-conditioned generation	Multiple extraction tasks can share one generative framework	Generation may omit, merge or hallucinate information
LLM based IE	Prompting, instruction tuning and structured outputs	Flexible domain adaptation and rich schemas	Needs strong admission rules, grounding and validation
Long context and hierarchical processing	Longformer, BigBird, RAPTOR	Processes or organizes information beyond short contexts	Additional computation and aggregation complexity
Retrieval augmented systems	DPR, RAG, GraphRAG	Connects generation to external evidence and private knowledge	Retrieval errors can propagate to the generated answer
Agentic systems	Planning, retrieval and tool execution	Can select different strategies for different questions	Requires explicit capability boundaries, state control and validation

2.14 Positioning of the DigiVoice Approach

The retained DigiVoice architecture does not attempt to introduce a new foundation model. Its contribution lies in combining and adapting several state-of-the-art principles to the constraints observed in real business calls.

The first principle is **schema-guided generative extraction**. Instead of producing

only free-form summaries, the model produces typed business objects according to an explicit contract. This follows the broader evolution toward generative and unified information extraction [12, 13, 15].

The second principle is **evidence grounding**. Extracted information is linked to transcript timestamps so that the source can be recovered and verified.

The third principle is **adaptive long-context processing**. We do not assume that the maximum model context window is the most reliable operating point. Instead, the transcript is segmented and independently processed before controlled reconciliation. This design is consistent with the limitations of long-context utilization reported in the literature [41].

The fourth principle is **separation between extraction truth and retrieval representation**. The typed Summary is the structured information layer, while the semantic summary is generated separately for retrieval. This prevents a lossy narrative representation from becoming the authoritative source for business facts.

The fifth principle is **deterministic validation around probabilistic models**. Pydantic validation, application-side post processing and Elasticsearch mappings establish deterministic interfaces around LLM outputs.

Finally, the question-answering layer separates intent understanding, planning, execution and answer composition. This follows the broader movement toward retrieval-augmented and tool-oriented language-model systems [37, 46, 47].

Figure 2.2 can be used to illustrate how these research directions converge in the project.

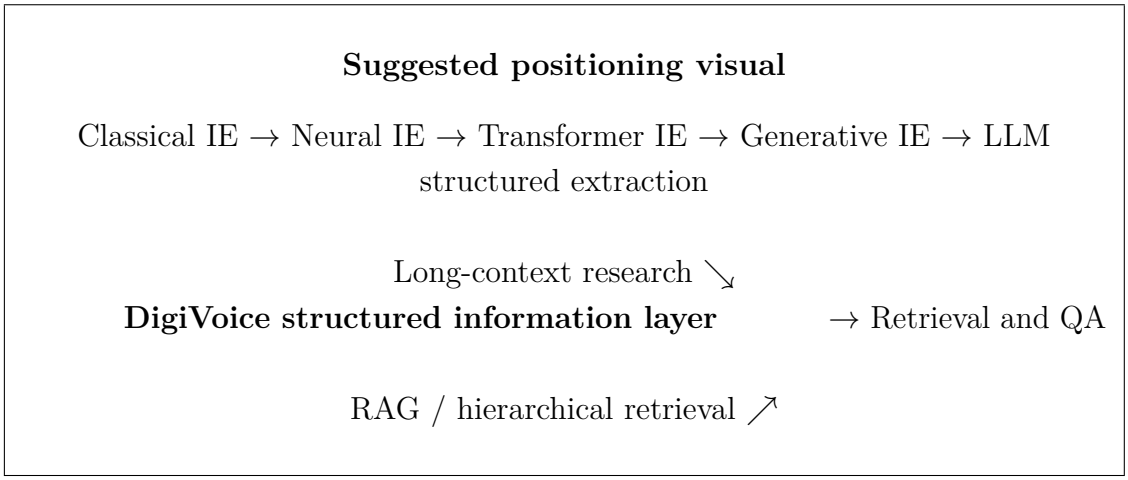


Figure 2.2: Positioning of the DigiVoice approach within the evolution of information extraction

2.15 State-of-the-Art Synthesis

The literature reveals a clear evolution in the definition of information extraction.

Initially, extraction focused on identifying a small number of predefined entities and relations. Neural methods then reduced manual feature engineering, while pretrained Transformers significantly improved contextual representations. Generative information extraction subsequently transformed the task from classification into schema-conditioned generation.

Large Language Models extend this flexibility but also make reliability a central concern. Their capacity to understand complex instructions enables rich business schemas, yet the same generative freedom can produce unsupported objects or inconsistent attributes. For long conversations, the challenge is further amplified by context length, information dispersion and conversational revision.

Modern systems therefore increasingly combine several mechanisms rather than depending on one model invocation: structured generation, segmentation, hierarchical processing, retrieval, validation, provenance and tool-based execution.

This observation strongly influenced the architecture adopted in this project. Rather than treating the LLM as the complete system, we use it as one probabilistic component inside a larger evidence-grounded pipeline. The structured Summary, controlled merge, deterministic post processing, searchable Elasticsearch representation and agentic query layer each address a different failure mode.

The resulting design can therefore be understood as the convergence of three research trajectories: generative information extraction, long-context processing and retrieval-grounded question answering.

Chapter 3

Real Call Audit and Data Understanding

3.1 Why we audited calls before finalizing a schema

The first mistake in an extraction project would be to design a schema from assumptions about sales calls. We instead reviewed real conversations. The first audit covered 15 calls and was then extended to 30 calls for a total duration of 20 hours and 32 minutes and after extended again to 50 calls. The purpose was to identify recurring interaction patterns, unreliable source fields and the types of facts that later questions would need.

3.2 Observed call purposes

The audited sample was dominated by technical scoping and qualification, but it also contained cost reconciliation, hybrid technical and cost reviews, onboarding, workshops, report reviews and annual claim planning. Table 3.1 reproduces the recorded distribution.

Table 3.1: Primary purposes in the 30 audited calls

Primary purpose	**Calls**
Technical scoping and qualification	18
Cost reconciliation	4
Hybrid technical and cost review	3
Onboarding and pre scoping	2
Workshop and training	1
Narrative and report review	1
Annual claim planning	1
Total	**30**

The distribution is an audit of the reviewed set. We do not treat it as an organization wide estimate of call frequency.

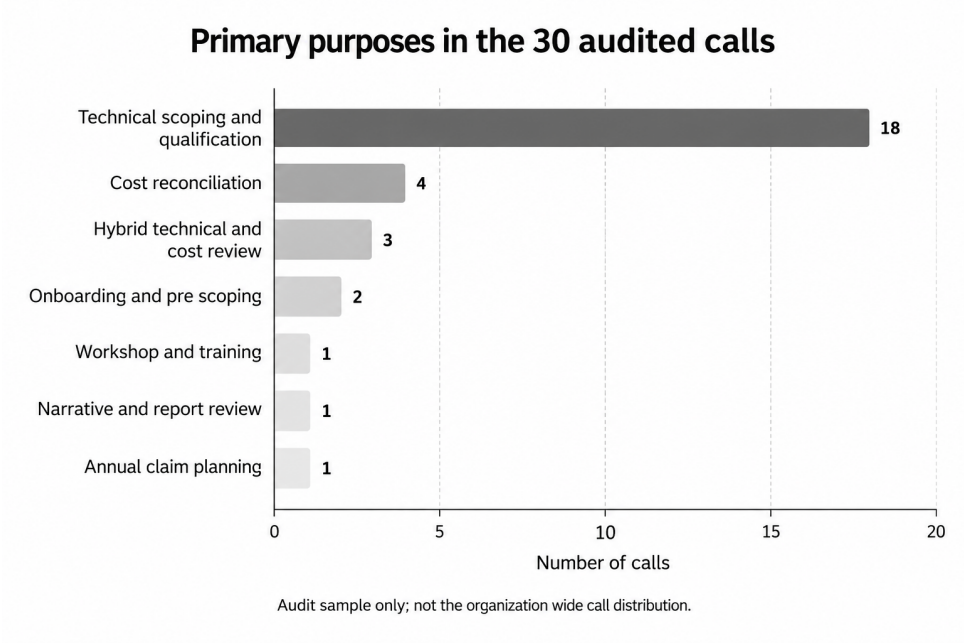


Figure 3.1: Horizontal bar chart of the seven audited call purposes using black, dark gray and light gray only.

3.3 Recurring business interaction pattern

Across technical scoping calls we repeatedly observed a sequence in which the client describes a project or feature, compares it with a previous system or normal method, explains where the baseline failed, provides experiments or evidence, reacts to the consultant's interpretation, reaches a provisional position and then provides additional documents or cost information before a final decision. Cost reconciliation calls follow a different path based on period, legal entity, workbooks, role allocation, project codes, subcontractor information and follow up.

This showed that the extraction contract could not be limited to entities and keywords. It needed to represent lifecycle, uncertainty, decisions, actions, documents and evidence.

3.4 Source reliability findings

The audit produced findings that later became prompt rules and post processing constraints.

Table 3.2: Audit findings and extraction implications

Observed finding	**Design implication**
No audited call had named speaker diarization. Generic labels such as client and salesman were used.	A speaker role cannot establish a named identity. A person mentioned in the transcript must not automatically be treated as an attendee.
Only 10 of 30 metadata SDR identities exactly matched the attendee identity found in the audit.	Metadata identity and transcript supported identity must remain distinguishable. A trusted configured SDR rule must be explicit rather than assumed globally.
The stored SCOPING category matched technical scoping content in only 11 of 16 calls carrying that label.	Stored categories cannot be used as unquestioned evaluation truth. Business purpose should be inferred from content when needed.
NON COMPLIANT contained several legitimate call purposes, while HANDOVER described a stage rather than a complete purpose.	Call purpose and process stage must be treated as different concepts.
Duplicate dialogue occurred and some call dates were repeated.	Facts must be deduplicated conservatively. Exact source timestamps and original date expressions must be preserved.

These findings are important because they explain why the final prompt contains many negative rules. The strictness was not added for style. It was introduced to block errors that were visible in the source data.

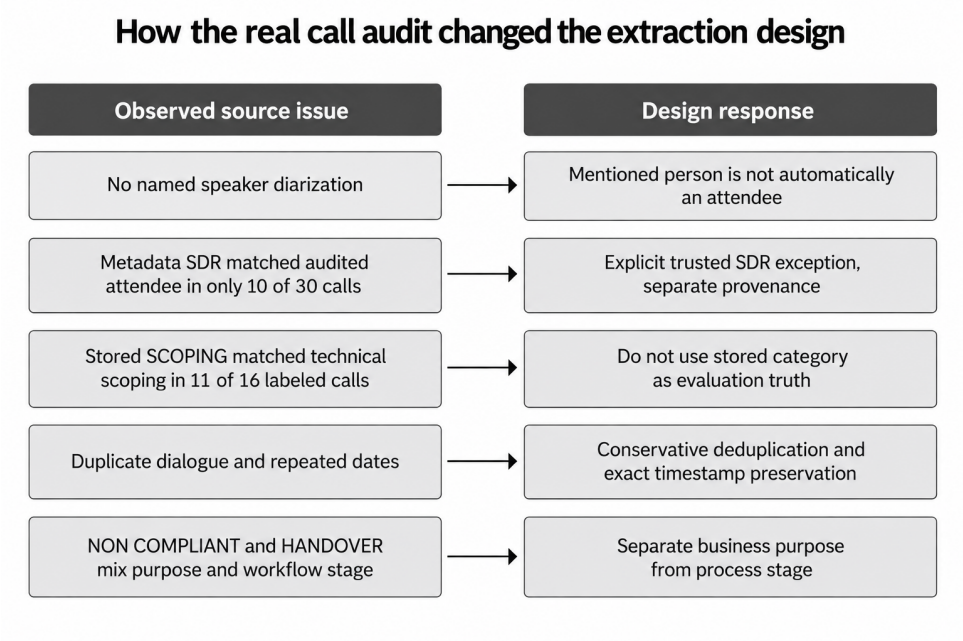


Figure 3.2: A two column cause and effect diagram connecting the main real call audit findings to the extraction rules they motivated.

Chapter 4

Business Questions and the Extraction Contract

4.1 From questions to information requirements

We gathered a foundation set of questions covering single call understanding, call quality, consultant performance, volume, recurring themes and client history. The questions include concerns, participants, project decisions, documents, next actions, scores, rankings, trends and cross call themes.

Instead of designing fields because they looked convenient, we mapped questions to the information that must survive ingestion.

Table 4.1: Representative business questions and required information

Question	Information that must be preserved
What concerns or blockers did the client raise?	Pain points, objections, uncertainties, impact and evidence.
What questions were asked and what remains unresolved?	Question text, asker when supported, answer when supported, resolution status and evidence.
What next steps were agreed, by whom and by when?	Decision, action, owner, status, time expression, timed contact and evidence.
Which projects were discussed and what happened to each?	Project identity, objective, period, status, technical challenge, eligibility state and project specific evidence.
Who was on the call and on which side?	Named people, side, role, attendance state and provenance.
Which documents were shared, requested or promised?	Document identity, type, lifecycle state, ownership when supported and evidence.
What information is reusable across calls?	Facts, topics, previous events, companies and normalized business states.

4.2 Contract terminology

We use the term extraction contract to mean the combination of a structured schema and the prompt rules that decide what is admitted into that schema. The model is only one element of the complete pipeline configuration. A pipeline configuration also includes the merge model, segmentation policy, evidence handling and post processing.

In the migration benchmark, **Legacy** is our label for the earlier schema and its associated prompt, while **Improved** is our label for the revised schema and prompt. These labels describe contracts. They do not describe model families and they do not imply quality by themselves.

4.3 The earlier contract

The earlier contract grouped information into participants and tagged names, organizations and projects, client needs and pain points, broad terms and timed contacts, documents and previous events, plus model generated metadata such as topics and decisions. It provided a useful first structured layer, but several business meanings shared broad fields and some state information was difficult to express explicitly.

Table 4.2: Earlier contract representation

Block	Representative fields	Purpose
People	participants, tagged names	Represent attendees and other named people.
Companies and projects	organization, companies, projects	Identify business context and discussed projects.
Needs and difficulties	client needs, client pain points	Preserve desired outcomes and expressed difficulties.
Information and follow up	terms, timed contacts	Store broad information and scheduled interactions.
Documents and history	documents, previous events	Preserve referenced artifacts and prior events.
Extracted metadata	topics, decisions	Provide additional queryable signals.

4.4 The current Summary contract

The current Summary contains 16 collections. Each collection exists because it answers a distinct retrieval need.

Table 4.3: Current Summary collections and semantic role

Collection	Semantic role
participants	Named human people with side, role, attendance and evidence.
companies mentioned	Material named organizations with an explicit relationship only when supported.
projects	Named projects with objective, period, status, technical context, eligibility and evidence.
topics	Main subjects of the conversation expressed as concise text labels.
needs	Client originated requirements or desired outcomes, with priority and status when supported.
pain_points	Supported client or project problems with category, impact oriented severity, status and evidence.
objections	Explicit resistance, constraint or reason not to proceed, separated from neutral facts.
service_interests	Client attitude toward a specific service, not merely a consultant explanation.
decisions	Supported decisions with explicit state.
actions	Tasks, commitments or proposals with ownership and lifecycle state.
timed_contacts	Scheduled or completed interactions with time information and state.
documents	Requested, promised, shared, received, reviewed or otherwise referenced documents.
previous_events	Historical events relevant to the current conversation.
facts	Reusable factual business information that deserves a dedicated representation.
open_questions	Questions with asker, answer and resolution state when supported.
uncertainties	Genuine unresolved ambiguity with possible interpretations when supported.

The schema is validated with Pydantic. Pydantic controls shape, types, enum values and structural constraints. It does not prove that the model extracted the correct facts. Factual quality requires a separate evaluation process.

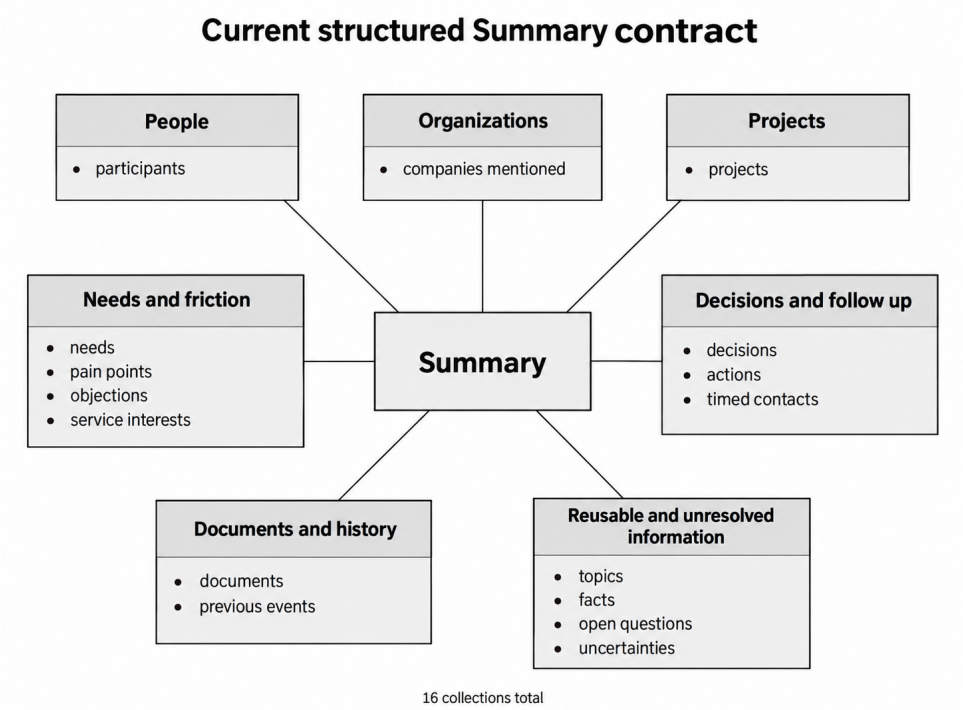


Figure 4.1: A monochrome schema map showing the 16 Summary collections grouped into people, organizations, projects, needs and friction, decisions and follow up, documents and history, reusable facts and unresolved information

Chapter 5

Prompt Engineering and Contract Evolution

5.1 Why prompt engineering became a core design problem

The experiments showed that a larger schema is not automatically a better extractor. The model must understand the admission boundary of every field. The practical lesson was that quality comes from the combination of schema, business definitions, evidence rules, negative rules and explicit abstention behavior.

The prompt therefore evolved from a general extraction instruction into a contract with field specific definitions, KEEP and REJECT examples, evidence rules, locality rules and merge rules.

5.2 Prompt milestones

Table 5.1: Main prompt and schema milestones

Stage	Change	Reason
V1 to V3	Increasingly explicit extraction rules, evidence requirements and positive and negative examples.	Reduce invented participants, broad semantic over extraction, duplicates and attribution mistakes.
V4	Additional prompt experiments with Luna and Terra.	Measure whether model behavior changes under a revised instruction set.
Legacy (V5)	Earlier Summary plus model generated metadata, tagged names and broad terms.	Provide a practical structured baseline.
V6	Richer business fields, explicit attendance and lifecycle states, lighter evidence output.	Make business meaning more queryable while controlling prompt output size.
V7	Summary only extraction, current schema and enums, timestamp evidence and stronger examples.	Separate model responsibilities from deterministic application logic.
Improved (V7.1)	Retain V7 architecture while strengthening semantic field definitions using the strongest useful V5 concepts.	Recover semantic precision without restoring obsolete fields or prompt bloat.

5.3 V7.1 design principle

The purpose of V7.1 was not to redesign the whole system. It was to preserve the architecture and schema of V7 while making field admission more precise. The project specification can be summarized as:

```
V7 architecture
+ V7 schema
+ V7 enums and statuses
+ V7 timestamp evidence
+ V7 examples
+ strongest useful V5 semantic definitions
without obsolete V5 fields
without model generated metadata
```

```
without duplicated rules
without prompt bloat
```

This is important because several observed errors were no longer failures to see information. They were failures to place valid information in the correct semantic field.

5.4 Evidence contract

The current extraction prompt minimizes evidence output from the model. For transcript evidence, the model emits the source and the exact raw timestamp. It does not reproduce transcript text or speaker text inside the model output. Post processing later resolves the readable evidence from the source call. This reduces duplicated output tokens and prevents the model from paraphrasing the evidence itself.

The core rule is that a transcript timestamp must appear exactly in the supplied transcript. It must not be normalized, rounded, translated or invented. Evidence should support the attached item and its populated attributes, not merely mention the same broad topic.

5.5 Participant admission rule

Participant identity was one of the most sensitive fields because the audit showed weak named diarization. We therefore apply a deliberately conservative attendance model.

1. A trusted configured SDR supplied by the runtime is treated as PRESENT under an explicit metadata exception.
2. Every other named human extracted from the transcript starts as MENTIONED NOT PRESENT.
3. The same person is upgraded to PRESENT only when positive current call evidence establishes participation.
4. Repetition, importance, role, company membership or action ownership do not prove attendance.
5. Generic speaker labels are never converted into names.

The label MENTIONED NOT PRESENT should be read as “attendance is not established by the current evidence.” It does not always prove physical absence.

5.6 Semantic admission gates

The prompt contains rules that prevent nearby concepts from collapsing into one another.

Table 5.2: Examples of current semantic admission gates

Candidate Information	Required treatment
A consultant asks the client to send a spreadsheet.	This can become an Action and a requested Document. It is not automatically a client Need.
A project is technically complex or uncertain.	Complexity alone does not make it a PainPoint and does not imply eligibility.
A client says a fee is too high to proceed.	This is an Objection. A neutral budget amount without resistance is a Fact.
A consultant explains a service.	This does not become ServiceInterest unless the client expresses a supported attitude toward that service.
Work happened historically.	Historical work is not automatically a current Action. Proposed, agreed and completed states must remain distinct.
A vague expression such as later or sometime appears.	It is not admitted as a TimedContact if there is no meaningful timing commitment.
A low level technical measurement appears.	It is not automatically a business Fact. Materiality is required.
A project was innovative, complex or experimental.	None of those properties alone establishes QUALIFIES or LIKELY eligibility.

5.7 Project eligibility discipline

The current prompt treats project eligibility as an explicit conclusion rather than an inference from technical sophistication. QUALIFIES requires an explicit positive conclusion. LIKELY requires a provisional positive conclusion. NEEDS_INFO requires an explicit need for additional information before deciding. DOES_NOT_QUALIFY requires an explicit negative conclusion. NOT_DISCUSSED is used when the project has no project specific eligibility assessment.

5.8 Chunk locality and boundary discipline

A chunk extractor is not allowed to reconstruct dialogue that lies outside its chunk. An isolated “yes” cannot create an agreement unless the relevant proposition is visible. A proposal without a client reaction cannot be marked accepted. A question whose answer is outside the chunk cannot be completed from memory. This locality rule is essential because independent chunk extraction only remains auditable if each local output is justified by local evidence.

5.9 Merge prompt discipline

The merge prompt is also intentionally constrained. The merger may reconcile duplicate objects and combine compatible attributes, but it is not allowed to perform fresh extraction from an unseen transcript. It must avoid moving a challenge or impact from one project to another, apply group level eligibility only when the referenced project set is unambiguous, attach later answers only to the correct question, and resolve aliases conservatively. When an identity remains uncertain, preserving uncertainty is preferred to guessing.

5.10 Separate semantic summary

We separated the semantic summary from extraction. The detailed Summary is the source of structured business facts. After it is validated and post processed, a dedicated request generates a short retrieval oriented paragraph. Earlier versions allowed a much longer representation; later versions were tightened toward approximately 70 to 100 words and a maximum around 900 characters. The goal is retrieval density, not a second call report.

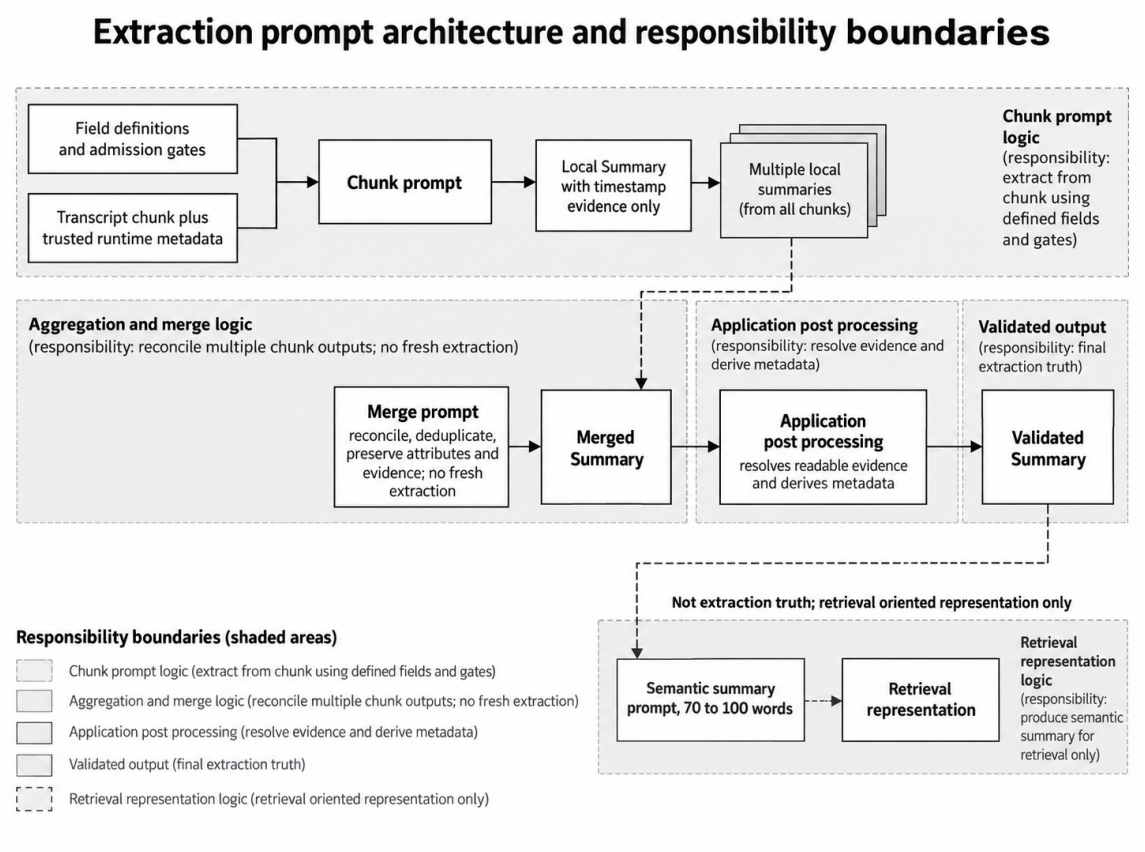


Figure 5.1: Prompt architecture diagram showing field contract, chunk task, merge task, timestamp evidence and separate semantic summary generation.

Chapter 6

Long Call Processing and Adaptive Chunking

6.1 Why segmentation became necessary

The audited call inputs vary strongly in size. The recorded cleaned JSON profile ranges from 9,854 tokens to 49,939 tokens, with an average of 23,341. Under the measured prompt configuration, the JSON plus prompt ranged from 11,938 to 52,023 tokens. The prompt overhead in that particular measurement was 2,084 tokens. It should not be treated as a universal constant because prompt versions change.

Table 6.1: Recorded input token profile

Statistic	Cleaned JSON	JSON plus prompt
Minimum	9,854	11,938
Average	23,341	25,425
Median	21,996	24,080
First quartile	16,577	18,661
Third quartile	27,548	29,632
90th percentile	33,220	35,304
95th percentile	41,279	43,363
Maximum observed	49,939	52,023

A fixed segmentation rule must therefore handle almost a five fold difference in call size. This motivated the move from one universal chunk limit toward an adaptive number of chunks.

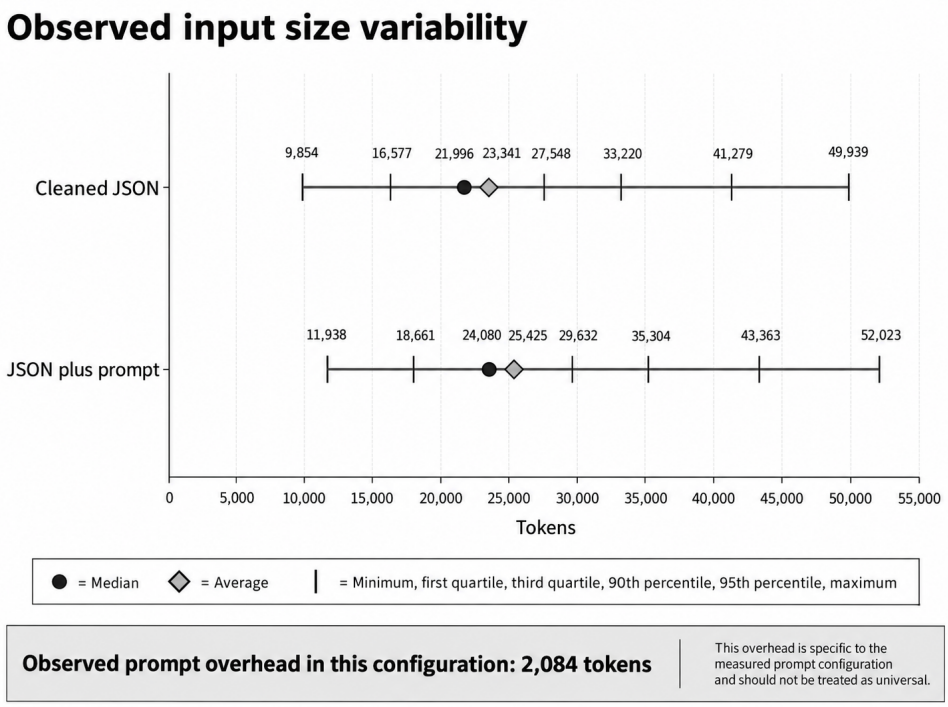


Figure 6.1: Range plot showing minimum, median, average and maximum input sizes for cleaned JSON and JSON plus prompt.

6.2 Whole call extraction as the baseline

Whole call extraction is operationally simple. It requires one extraction request and avoids cross chunk reconciliation. However, the experiments showed that whole call processing can miss information on some calls, while smaller segments can increase repeated context, API requests and duplication. Whole call processing therefore remains a baseline rather than the universal strategy.

6.3 Fixed chunk experiments

On the short call of approximately 9.8k tokens under V3, Luna scored approximately 77.2% with 5k segmentation and 66.0% when the call was processed as one full input. Terra moved in the opposite direction, scoring approximately 65.3% at 5k and 71.4% on the full call. This result showed that segmentation interacts with the model and cannot be optimized from call length alone.

On the longer Luna V3 sweep, the recorded audited quality values were 59.25% at 5k, 72.35% at 8k, 76.86% at 10k, 73.91% at 20k and 65.59% on the whole call. The best point in that sweep was in the middle rather than at either extreme.

Table 6.2: Long call Luna V3 chunk sweep

Processing setting	Audited quality
5k	59.25%
8k	72.35%
10k	76.86%
20k	73.91%
Whole call	65.59%

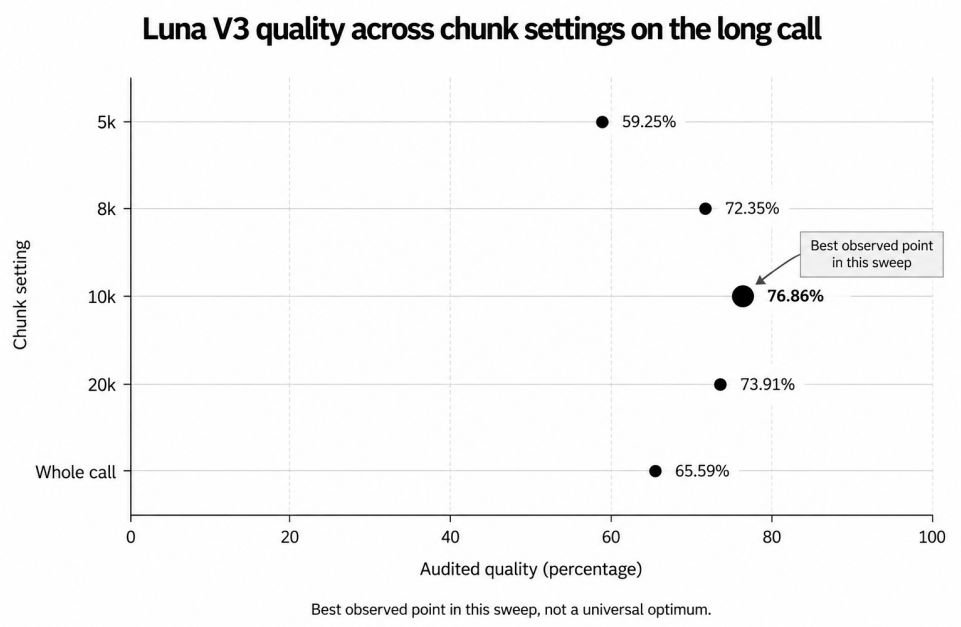


Figure 6.2: Line chart or horizontal dot plot of the five Luna V3 chunk sweep quality values.

6.4 Retained adaptive heuristic

The current operating heuristic targets a small number of chunks based on transcript length.

Table 6.3: Adaptive target chunk count

Transcript length	Target chunks
Less than 10k tokens	2
From 10k to less than 25k tokens	3
From 25k to less than 40k tokens	4
40k tokens or more	5

The segmentation implementation must also preserve turn boundaries and account for overlap. The current Transform implementation uses configurable overlap turns and applies the extraction prompt independently to each chunk before the merge.

Likely interpretation: the heuristic is a pragmatic balance between local context, repeated prompt overhead and merge complexity.

Chapter 7

Model, Prompt and Processing Benchmarks

7.1 Initial formal benchmark

The first formal manager oriented benchmark used one 87.7 minute call with 36,039 transcript tokens. We compared two models, two prompt versions and three chunk limits, giving 12 configurations with one run per configuration. The scorecard measured audited quality, end to end latency, API request count, token usage, estimated direct cost and structural behavior.

The audited quality composite in that benchmark assigned 20% to people, 15% to entities, 27% to business facts, 18% to decisions and topics, 12% to derived metadata and 8% to structure and evidence. This historical quality measure is specific to that experiment and should not be equated with the later common information capture metric.

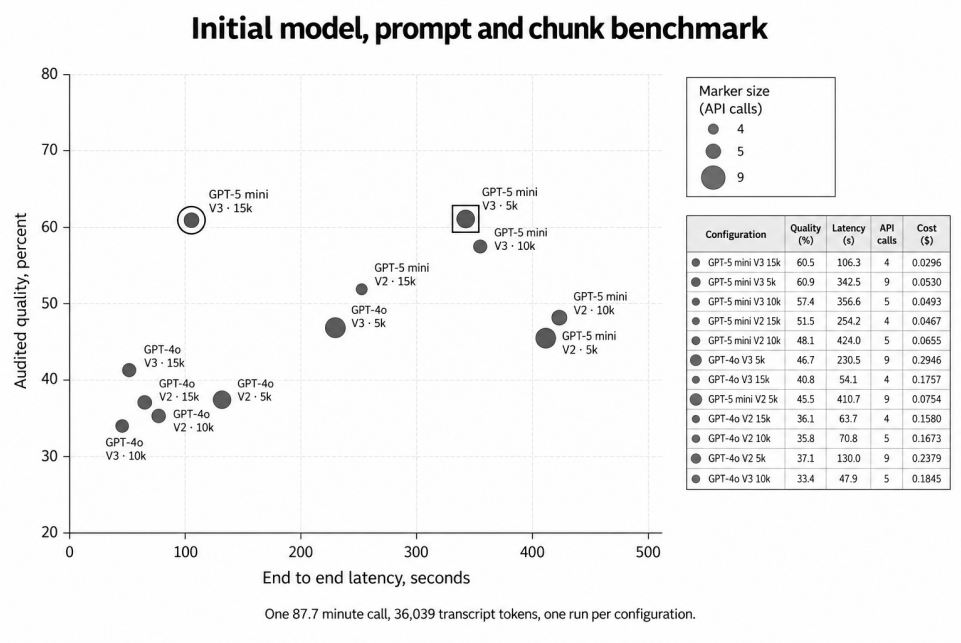
Table 7.1: Complete 12 configuration initial benchmark

Configuration	Quality	Latency s	Calls	Tokens	Cost USD
GPT 5 mini V3 15k	60.5%	106.3	4	71,685	0.0296
GPT 5 mini V3 5k	60.9%	342.5	9	124,366	0.0530
GPT 5 mini V3 10k	57.4%	356.6	5	90,090	0.0493
GPT 5 mini V2 15k	51.5%	254.2	4	75,124	0.0467
GPT 5 mini V2 10k	48.1%	424.0	5	84,825	0.0655
GPT 4o V3 5k	46.7%	230.5	9	103,720	0.2946
GPT 4o V3 15k	40.8%	54.1	4	63,118	0.1757
GPT 5 mini V2 5k	45.5%	410.7	9	109,277	0.0754
GPT 4o V2 15k	36.1%	63.7	4	50,341	0.1580
GPT 4o V2 10k	35.8%	70.8	5	53,535	0.1673
GPT 4o V2 5k	37.1%	130.0	9	73,546	0.2379
GPT 4o V3 10k	33.4%	47.9	5	68,971	0.1845

The highest raw quality was 60.9% for GPT 5 mini V3 at 5k. The 15k version scored 60.5%, only 0.4 percentage points lower, while reducing latency from 342.5 seconds to 106.3 seconds and estimated direct cost from 0.0530 to 0.0296 dollars. This is approximately 69 less latency and 44 lower cost on that call. We therefore selected GPT 5 mini V3 at 15k

as the initial operating point.

this was one run on one call. It supported a local engineering decision, not a statistically reliable model ranking.



Configuration	Quality (%)	Latency (s)	API calls	Cost ($)
GPT-5 mini V3 15k	60.5	106.3	4	0.0296
GPT-5 mini V3 5k	60.9	342.5	9	0.0530
GPT-5 mini V3 10k	57.4	356.6	5	0.0493
GPT-5 mini V2 15k	51.5	254.2	4	0.0467
GPT-5 mini V2 10k	48.1	424.0	5	0.0655
GPT-4o V3 5k	46.7	230.5	9	0.2946
GPT-4o V3 15k	40.8	54.1	4	0.1757
GPT-5 mini V2 5k	45.5	410.7	9	0.0754
GPT-4o V2 15k	36.1	63.7	4	0.1580
GPT-4o V2 10k	35.8	70.8	5	0.1673
GPT-4o V2 5k	37.1	130.0	9	0.2379
GPT-4o V3 10k	33.4	47.9	5	0.1845

Figure 7.1: Scatter plot of all 12 configurations with latency on the horizontal axis and audited quality on the vertical axis, with labels for every configuration.

7.2 Why the benchmark did not end the investigation

Field level inspection exposed a crucial weakness in the aggregate score. On the selected GPT 5 mini V3 15k run, all four expected projects were recovered and all eight expected participants were recovered, but three additional participant entries were produced. At the same time, zero of six expected decisions and zero of eight expected broad terms were recovered in the inspected scorecard.

This taught us that a global percentage can hide catastrophic failure in an important field. From that point onward, field coverage, unsupported additions and evidence validity became first class diagnostics.

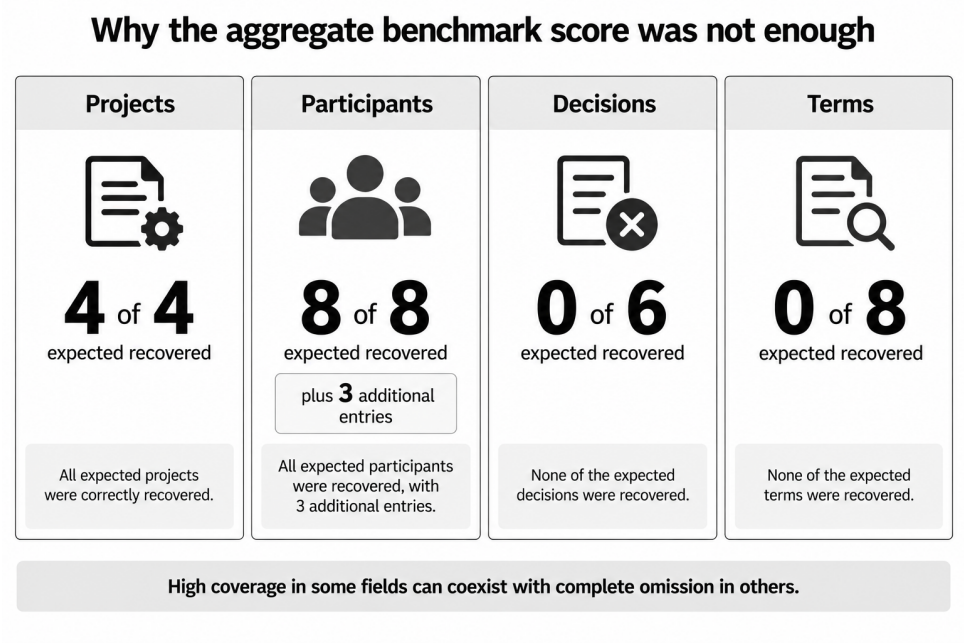


Figure 7.2: Four panel diagnostic showing projects 4 of 4, participants 8 of 8 with 3 extra, decisions 0 of 6 and terms 0 of 8.

7.3 Luna and Terra long call comparison

We then tested Luna and Terra under the same labeled V3 and 10k setting on a long call. Luna recorded 76.86 audited quality, 156.5 seconds latency, five requests and an estimated run cost of 0.05224 dollars. Terra recorded 72.71 142.6 seconds, five requests and 0.32495 dollars. Luna was therefore the more attractive candidate on quality and estimated cost in this setting, while Terra was slightly faster.

Table 7.2: Long call Luna and Terra comparison under V3 and 10k

Model	Quality	Latency s	Requests	Cost USD
Luna	76.86%	156.5	5	0.05224
Terra	72.71%	142.6	5	0.32495

7.4 Short call V3 and V4 checks

The short call experiments demonstrated that the result can change with prompt version and segmentation. Under V3, Luna benefited from 5k segmentation while Terra performed better on the full call. Under a later V4 4k comparison, Terra reached a higher recorded quality than Luna at a substantially higher estimated cost.

Table 7.3: Recorded short call comparisons

Configuration	Quality	Latency s	Cost USD
Luna V3 5k	77.2%	32.5	0.0141
Luna V3 full call	66.0%	13.8	0.0075
Terra V3 5k	65.3%	40.7	0.0938
Terra V3 full call	71.4%	10.4	0.0423
Luna V4 4k	76.5%	24.6	0.0148
Terra V4 4k	81.0%	24.7	0.1080

We do not attribute the difference between a V3 5k run and a V4 4k run to the prompt alone because two independent variables changed. This is an important experimental discipline in the report: model, prompt, segmentation and merge configuration must be identified separately.

Chapter 8

Extraction Strategy and Controlled Merge

8.1 Strategies that we tested

Chunking solved an input size problem but created a new consistency problem. A business fact can be spread across several chunks. A person can be named later. A proposal can appear in one chunk and its acceptance in another. We therefore compared several strategies for accumulating information.

Table 8.1: Processing strategies and retained decision

Approach	**Advantage**	**Observed risk or limitation**	**Decision**
Whole call extraction	Simplest baseline and one extraction request.	Limited intermediate visibility and omissions on some long call settings.	Baseline
Fixed segmentation	Bounds input size.	Can split related facts, increase repeated prompt cost and create duplicates.	Interim
Cumulative extraction	Progressively updates one shared state.	Later steps can delete facts, replace attributes with null or regress a status.	Rejected as main path
Guarded snapshots and replay	Protects updates and allows reconciliation.	Adds sequential complexity and did not establish enough quality advantage.	Experiment
Independent extraction and controlled merge	Every chunk output is inspectable, retryable and potentially parallelizable.	Requires careful identity resolution, deduplication and state preservation.	Retained

8.2 Why independent extraction was retained

Independent extraction separates two responsibilities. The chunk stage asks what information is supported locally. The merge stage asks how several already extracted objects

should be reconciled. This makes failure diagnosis easier because we can identify whether information was missed during extraction or lost during merge.

The Transform implementation reflects this separation. It processes each chunk through the structured Summary schema, stores each parsed chunk result, then sends the list of chunk results to the merge step.

8.3 Merge invariants

The merge must preserve several invariants.

1. Deduplicate repeated objects while retaining supported attributes.
2. Resolve aliases only when the source supports identity equivalence.
3. Preserve exact evidence timestamps and provenance.
4. Preserve lifecycle changes such as proposed, agreed and completed.
5. Keep project evidence attached to the correct project.
6. Link a response to a question or proposal only when the relationship is supported.

A smaller object count after merge can be correct when duplicates are removed. It becomes a merge failure when a supported attribute, state or evidence pointer disappears.

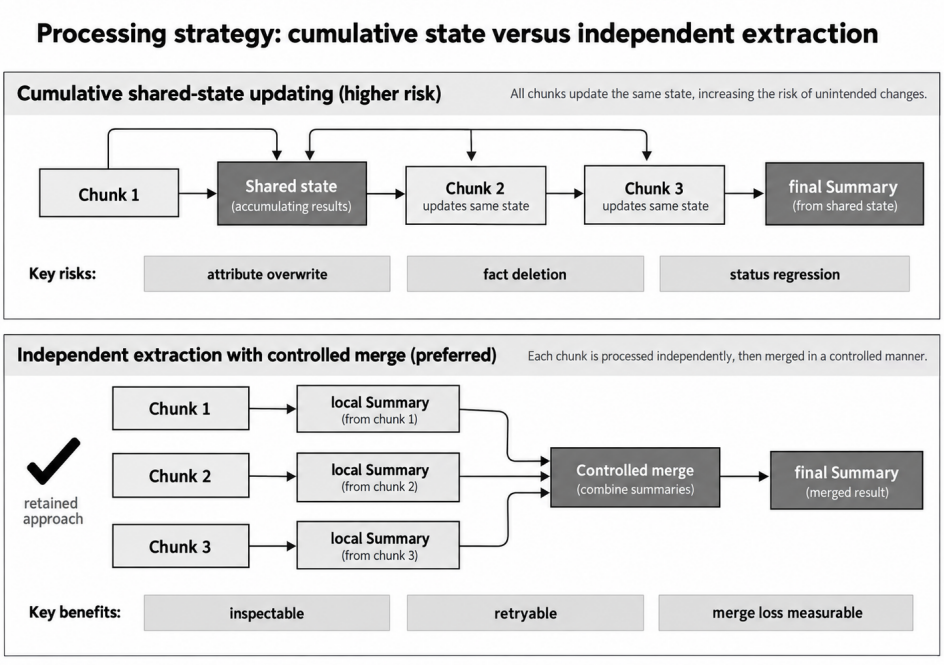


Figure 8.1: Two lane comparison: cumulative shared state versus independent chunk extraction followed by controlled merge. Emphasize the retained independent path

8.4 Model pairing for extraction and merge

We explored Luna to Luna, Terra to Terra and Luna to Terra pairings. The project retained Luna for both extraction and merge. However, the current source register does not contain a numerical pair scorecard that proves the margin between these three pairings. The presentation also notes that the quantified comparison still needs to be attached for complete traceability.

operating choice: Luna to Luna is the reported retained configuration.
evidence gap: its numerical superiority as a two stage pair is not established by the currently identified artifacts. We therefore do not invent a percentage advantage.

A controlled future pair benchmark should reuse identical saved Luna chunk outputs when comparing Luna merge versus Terra merge. That isolates the merger effect. Terra to Terra changes both extraction and merge and therefore measures the entire configuration.

Chapter 9

Post Processing, Metadata and Elasticsearch Representation

9.1 Why the model does not own every output field

The current design deliberately removes deterministic work from the language model. The model extracts business information into the Summary. Application code then resolves readable evidence, copies trusted source metadata and evaluation data, derives counts, lists and boolean flags, and creates the final stored representation. This separation reduces cost and prevents the model from rewriting information that already exists in the source.

9.2 Three information layers used for querying

The agent architecture distinguishes the following logical areas inside a call record.

Table 9.1: Logical call information layers

Layer	Role	Examples
Metadata	Fast filtering, grouping and scalar aggregation.	SDR identity, business unit, call type, call date, final score, derived presence flags, project names, companies, status flags, semantic summary.
Summary	Business content with item level evidence.	Participants, projects, pain points, objections, decisions, actions, documents, facts, open questions and uncertainties.
Evaluation	DigiVoice scoring and explanation data copied from the source.	Criteria, subcriteria, scores, justifications and quoted proof when present.

The raw conversation remains important for evidence recovery and verification. The presentation explicitly keeps access to transcripts as part of the question answering vision.

9.3 Derived metadata as a cost control mechanism

If a question only asks whether a call contains objections, the system should not scan every nested objection object merely to learn that at least one exists. We therefore derive simple metadata flags from the validated Summary. Likewise, final score aggregation belongs to a numeric metadata field rather than a language model read over evaluation explanations.

This design turns ingestion cost into cheaper repeated queries. The extraction pipeline performs expensive semantic work once; many later questions can use deterministic Elasticsearch filters and aggregations.

9.4 Semantic summary as a retrieval representation

The semantic summary is stored for meaning oriented retrieval, but it is not the source of truth for extraction. It is generated after validation from already accepted information. This distinction reduces the chance that a fluent paragraph introduces a fact that was never admitted by the structured contract.

Chapter 10

Evaluation Methodology and Ground Truth

10.1 Why the original evaluation approach was not enough

When the schema changes, a direct field by field comparison becomes unfair. Legacy and Improved do not have identical fields or identical admission rules. A ground truth built without reading the extraction prompt can also reward information that the evaluated contract was never instructed to extract.

We therefore rebuilt the references from the raw call while considering the exact prompt and schema associated with each version. The result is a prompt aware reference for each contract.

10.2 Three evaluation phases

1. Legacy output versus the Legacy prompt aware reference. This asks whether Legacy respects its own contract.
2. Improved output versus the Improved prompt aware reference. This asks whether Improved respects its own contract.
3. Legacy and Improved outputs versus the same common information ledger. This asks which output preserves more of the same business information.

The third phase is schema agnostic in the sense that a business information unit can be represented in different native fields. It is not free from annotation judgment. The ledger still depends on a clearly defined unit and matching rubric.

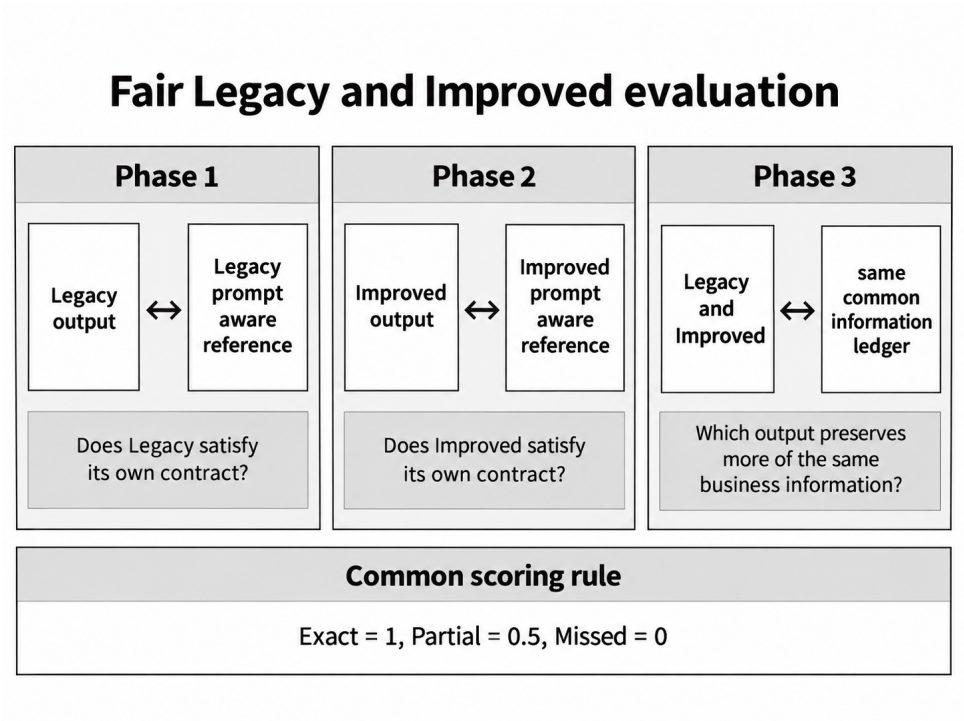


Figure 10.1: Three phase evaluation diagram with Legacy native evaluation, Improved native evaluation and shared common information comparison.

10.3 Soft and strict matching

For the migration notebooks, an exact match receives one unit of credit, a partial semantic match receives one half unit and a missed unit receives zero.

Let E be the number of exact units, P the number of partial units and N the number of expected units. The weighted true positive credit is

$$TP_{soft} = E + 0.5P. \tag{10.1}$$

Soft recall is

$$R_{soft} = \frac{TP_{soft}}{N}. \tag{10.2}$$

When explicitly recorded unsupported additions are available as FP, soft precision is

$$P_{soft} = \frac{TP_{soft}}{TP_{soft} + FP}. \tag{10.3}$$

Soft F1 is the harmonic mean

$$F1_{soft} = 2\frac{P_{soft}R_{soft}}{P_{soft} + R_{soft}}. \tag{10.4}$$

Strict capture removes partial credit. Only exact units receive credit. These two views

answer different questions. Soft matching measures meaningful partial preservation, while strict matching measures exact compliance with the annotation identity and attributes.

10.4 Native contract scores

The two prompt aware notebooks report the following native soft F1 values.

Table 10.1: Native prompt aware contract scores

Call	Legacy units	Legacy F1	Improved units	Improved F1
6830	75	90.7%	61	92.9%
7517	51	68.2%	44	85.0%

We do not subtract these values to claim a direct schema gain because the denominators and contracts differ.

10.5 Common information soft capture

For the shared ledger, both outputs use the same denominator.

$$\text{Soft Capture} = 100 \times \frac{E + 0.5P}{N}. \tag{10.5}$$

Table 10.2: Common information soft capture

Call	Units	Legacy credit	Improved credit	Legacy	Improved	Gain
6830	45	36.0	40.5	80.0%	90.0%	10.0 pp
7517	30	16.5	22.0	55.0%	73.3%	18.3 pp
Mean				67.5%	81.7%	14.2 pp

If all 75 common units are pooled instead of weighting calls equally, Legacy captures 70.0 and Improved captures 83.3 a gain of 13.3 percentage points. These aggregation methods answer slightly different questions. Neither should be generalized from two calls to the production corpus.

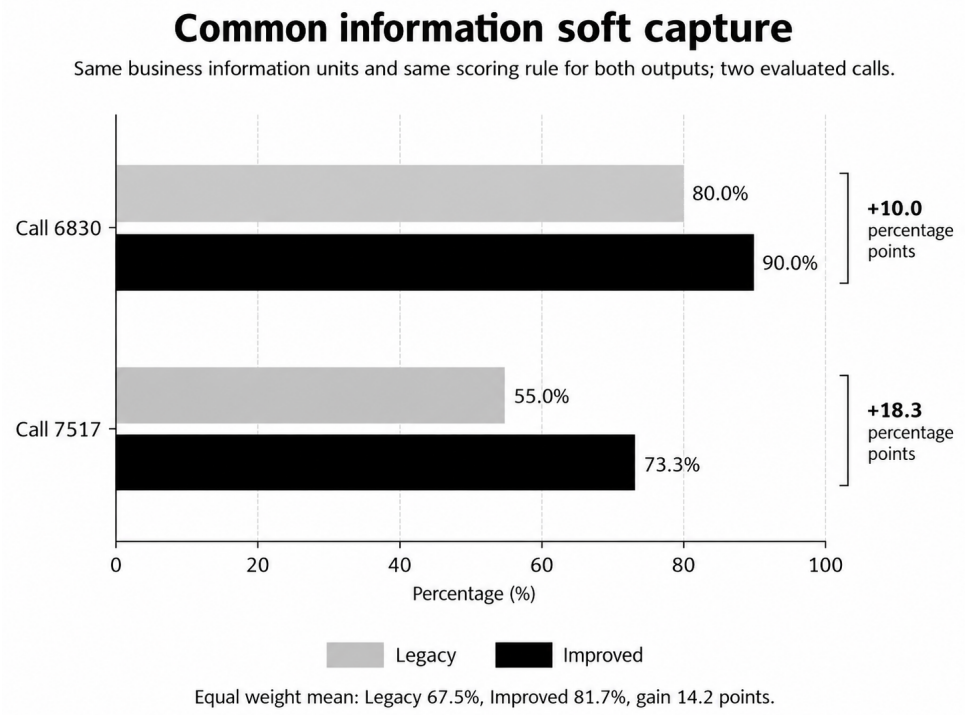


Figure 10.2: Paired horizontal bars for call 6830 and call 7517 showing Legacy versus Improved common information capture on the same 0 to 100% scale.

10.6 Category gains and regressions

The average gain does not mean every category improved. The common ledgers reveal both progress and regressions.

Table 10.3: Selected category changes in the two common ledgers

Call and category	Legacy	Improved	Change
6830 Projects	81.3%	100.0%	18.8 pp
6830 Problems	60.0%	90.0%	30.0 pp
7517 Rules and facts	0.0%	100.0%	100.0 pp
6830 People	100.0%	66.7%	negative 33.3 pp
6830 Decisions	100.0%	75.0%	negative 25.0 pp
6830 Rules and facts	87.5%	75.0%	negative 12.5 pp
7517 Historical events	100.0%	50.0%	negative 50.0 pp

This is why we retained field level error analysis after the headline score. Some large changes also come from very small category denominators, so category percentages should always be read with unit counts in the notebooks.

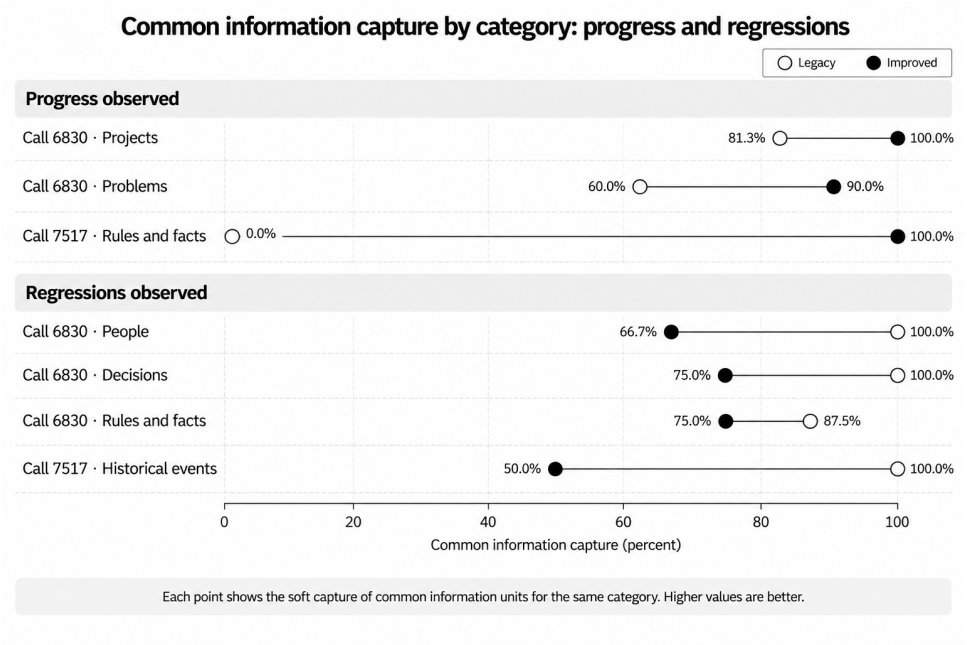


Figure 10.3: Dot comparison chart grouped into progress and regressions for projects, problems, rules and facts, people, decisions and historical events.

10.7 Gold reference reproducibility package

We later strengthened reproducibility by packaging a gold reference with source hashes, schema hash, prompt hashes, annotation ledger and validation metadata. One packaged reference for call 6830 records 571 evidence references, seven duplicate timestamps, 61 validation checks and 11 ambiguous attributes automatically excluded from extraction accuracy.

The same manifest freezes SHA 256 fingerprints for the chunk prompt, merge prompt and semantic summary prompt. This matters because a benchmark is only reproducible when the exact extraction contract can be identified. Reusing a cached extraction after changing a prompt would otherwise compare different systems under the same label.

Chapter 11

Retained Extraction Pipeline and Cost Decision

11.1 Current processing flow

The retained flow has five distinct responsibilities.

1. Prepare the source call, count the input and segment the conversation adaptively while respecting turn boundaries.
2. Extract a Summary independently from each chunk using the current prompt contract.
3. Merge chunk outputs using a controlled reconciliation prompt.
4. Validate the structure with Pydantic, resolve evidence from source timestamps, copy trusted source metadata and evaluation information, and derive searchable metadata.
5. Generate a separate compact semantic summary for retrieval.

This design keeps the structured Summary as the main business representation and avoids asking the model to regenerate data that the application can derive deterministically.

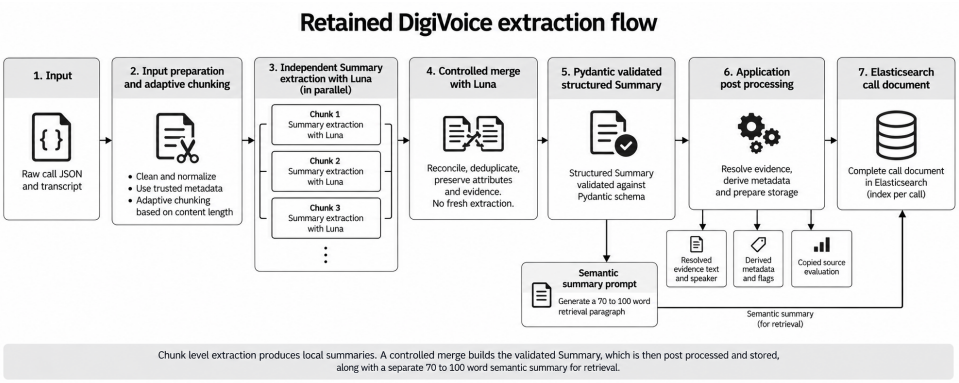


Figure 11.1: End to end extraction flow: source call, adaptive chunks, Luna extraction, Luna merge, validated Summary, application processing, Elasticsearch document and separate semantic summary request.

11.2 Reported batch cost

Table 11.1: Reported batch processing cost

Basis	Legacy	Improved	Difference
50 calls	0.87 USD	1.00 USD	0.13 USD
Per call average	0.0174 USD	0.0200 USD	0.0026 USD
Relative batch cost			14.9%

The batch figures are different from the historical single call benchmark cost estimates. They should not be mixed into one cost scale unless the included processing stages, pricing date, retries and semantic summary requests are documented consistently.

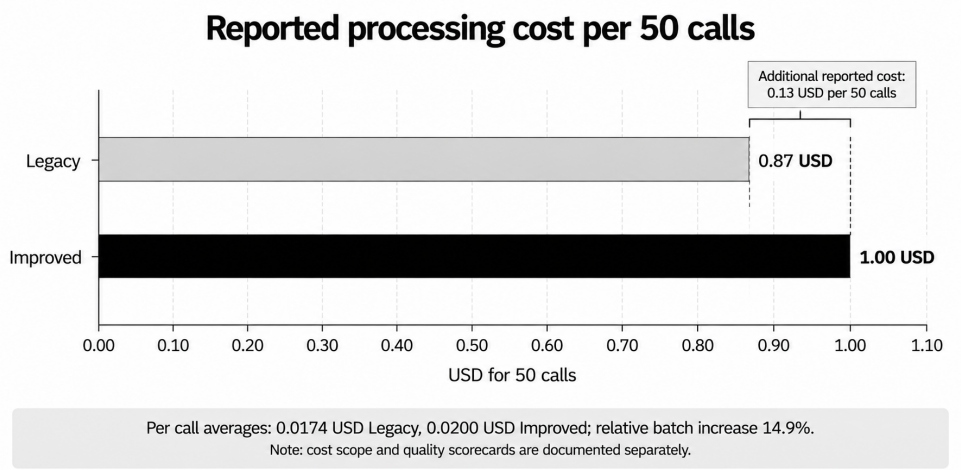


Figure 11.2: Two horizontal bars for reported cost per 50 calls: Legacy 0.87 USD and Improved 1.00 USD, with a small annotation for the 0.13 USD difference.

11.3 Decision statement

The evidence from the two migration calls supports retaining Improved because it captured more of the shared business information under the common ledger, while the reported incremental batch cost was small in absolute terms. The decision remains bounded by the evaluation sample. A production release benchmark should use a representative fixed call set and should record field errors, exact prompt fingerprints, end to end latency and complete cost scope.

Chapter 12

From Structured Calls to a Question Answering Agent

12.1 Why an agent layer was necessary

A well extracted call database does not by itself answer natural language questions. User requests can require different operations. Some ask for one call. Some require filtering several calls. Some require aggregation. Some need evaluation explanations. Some depend on previous turns. Some contain ambiguous names or incomplete references.

We therefore designed an agent that translates the user question into a controlled execution process rather than sending the question directly to Elasticsearch or directly to a general language model.

12.2 Architecture shown in the project presentation

The presentation organizes the system into a User, Intent Classifier, Planner and Router, execution tools, Call Summaries Database, State Management, Answer Composer and Final Answer. The execution tools include retrieval, aggregation, context use and general information handling. State is shown through Query State, Last Messages State and Router Steps State.

The architecture answers a sequence of concrete product questions.

Table 12.1: Architecture components and the question each component answers

Component	Question it answers
Intent Classifier	What kind of request is this and what shape of answer does the user expect?
Planner and Router	What retrieval or aggregation operation must be performed, with what scope?
Execution tools	How do we perform the required read, search, filter or aggregation?
Call Summaries Database	Where is the structured call information and supporting source data stored?
Query State	Which explicit scope should remain active during a conversation?
Last Messages State	Does the current request depend on the previous question and answer?
Router Steps State	What has already been planned and executed for the current request?
Answer Composer	How do we convert retrieved evidence into a concise answer that matches the requested format?

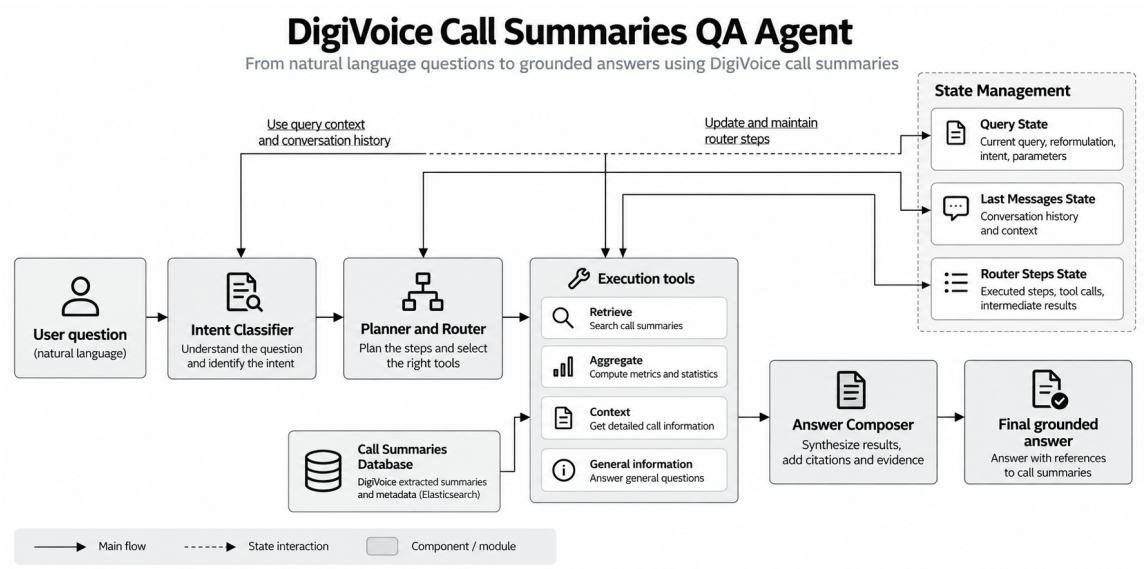


Figure 12.1: Monochrome version of the presentation architecture: User, Intent Classifier, Planner and Router, execution tools, call database, state management, Answer Composer and Final Answer.

12.3 Earlier end to end baseline

An earlier Architecture document describes an implemented pipeline with an intent classifier, session state, session memory, router, brain dispatcher, structured routes, fallback cascade, answer generator and API layer. The project file map includes modules for extraction, transformation, loading, intent classification, routing, execution, pipeline orchestration and tests, plus FastAPI endpoints for query, streaming, sessions and ingestion.

This baseline demonstrates that the project progressed beyond a diagram. It provided an end to end shape that could be exercised in the demo and revealed practical issues such as wrong aggregation metrics, stale filters and fallback behavior after zero hits.

12.4 Why the agent architecture evolved again

The earlier classifier carried many routing specific signals. We later simplified the contract so classification would remain a semantic understanding layer rather than a hidden Router. The current minimal QueryIntent has only four fields: a standalone reformulated question, a high level intent, an expected answer type and a confidence value.

This simplification creates a cleaner boundary but also means the Router must perform more detailed planning itself. A later architecture review therefore replaced the older route table dependency with a hybrid Router design.

Chapter 13

Architecture

13.1 Intent Classification

13.1.1 Current four field contract

The classifier receives the current user question and, when available, the previous question and answer. It outputs a typed QueryIntent.

```
QueryIntent
    reformulated_question : required string
    intent                : high level semantic area
    answer_type           : expected final answer shape
    confidence            : float from 0 to 1
```

The classifier does not answer the user, generate Elasticsearch queries, choose fields, create filters, select a route or build a physical query plan. Its main responsibility is to preserve user intent while resolving references that can safely be resolved from the previous turn.

13.1.2 Intent categories and answer shapes

The reviewed classifier implementation uses semantic areas such as call content, consultant performance, volume trends, content themes, client history and a general fallback. The prompt also contains a call quality category, while one inspected Python type omitted that value. This mismatch was identified during Router review and must be resolved so the prompt and Pydantic contract describe the same enum.

The answer type is one of single value, list, ranking, comparison, narrative, yes or no, and trend. The answer type controls presentation expectations but does not decide the retrieval mechanism.

13.1.3 Follow up resolution

The classifier uses the previous exchange to rewrite implicit questions. For example, after "How did Omar perform last month?", the user may ask "And Sarah?". The classifier should reformulate this into "How did Sarah perform last month?" without requiring the Router to interpret the pronoun like fragment from raw chat history.

When a reference cannot be safely resolved, the classifier keeps the wording faithful and lowers confidence. Low confidence then becomes an input to clarification policy downstream.

13.1.4 Interface issues discovered during review

The architecture review found two concrete boundary issues in the inspected classifier snapshot.

1. The prompt listed a call quality intent that was missing from the Python Subject literal in that snapshot.
2. The classify method assigned `original_question` onto a model configured with `extra="forbid"`, even though that field was not declared.

These are not conceptual failures of the classifier. They are contract consistency bugs. The clean fix is to align the enum and keep the raw original question in the pipeline trace rather than adding undeclared data to QueryIntent.

13.2 Router, Planning and Execution

13.2.1 Router responsibility

The latest Router design receives the four field QueryIntent and turns the preserved business question into explicit retrieval instructions. It is responsible for interpreting concepts, resolving entities, normalizing dates, selecting a supported operation, choosing the smallest useful projection and rejecting or clarifying requests that cannot be planned safely.

The Router should not execute Elasticsearch, answer the business question, invent an index field, infer a client from a mere company mention or silently choose an unspecified business metric.

13.2.2 Hybrid Router design

The latest specification recommends one constrained semantic planning pass followed by deterministic binding and validation.

1. Confidence gate checks whether the question can be interpreted safely.
2. Schema constrained semantic planner identifies requested business concepts, metrics and relationships.
3. Date normalizer resolves expressions such as this quarter against a supplied clock and timezone.

4. Entity resolver maps names to typed canonical values and surfaces ambiguity.
5. Capability compiler binds logical concepts to verified data capabilities.
6. Plan validator rejects unknown fields, incompatible operations and incomplete plans.
7. Brain or Executor generates the physical Elasticsearch query and runs it within trusted access boundaries.

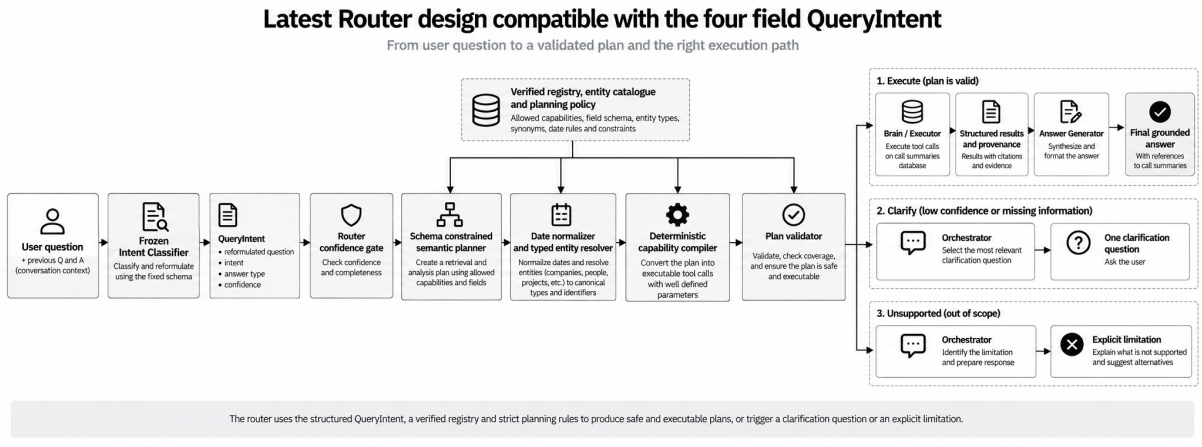


Figure 13.1: Latest Router design from four field QueryIntent through confidence gate, semantic planner, entity and date resolution, capability compiler, plan validator, Executor and grounded Answer Generator.

13.2.3 Why planning uses business operations rather than raw DSL

The Router should plan in a small vocabulary such as read, aggregate and search. The Executor owns physical Elasticsearch syntax. This separation makes plans testable without binding prompt output to index implementation details and reduces the chance that a language model invents raw field names.

The first version of the revised Router intentionally constrains a plan to one typed step. General multi step dependency references are not yet advertised. A request that truly requires the result of one step to parameterize another should be explicitly unsupported until typed references are implemented.

13.2.4 Representative planning cases

Table 13.1: Representative Router challenges

Question	Planning requirement
What are Acme's pain points?	Resolve Acme as the correct entity type, select pain point content and potentially retrieve across several calls.
Top 5 SDRs by average score this quarter.	Resolve time bounds, compute average final score, group by SDR, sort descending and limit to five.
Which calls had objections but no next step?	Combine a positive objections condition with a negative next step condition using a validated definition of next step.
Why did call X receive a low score?	Retrieve evaluation criteria and explanations, not only the final score.
What projects have we discussed with Acme?	Filter the client scope, read projects from several calls and deduplicate in the result layer.
How has call volume changed over the last six months?	Count calls in time buckets and return chronological values.
Compare Omar and Sarah this quarter.	The word compare does not define a metric. The system should clarify unless a product approved comparison profile is explicit.
Find calls where the client was uncertain whether the project qualified.	Distinguish client uncertainty from extraction uncertainty and potentially use a validated semantic search profile.
What are X's pain points?	Resolve X against a typed catalog or ask for clarification when the name is ambiguous.
How is he doing? with confidence 0.35	Do not plan blindly. Return a clarification decision through the orchestrator.

13.2.5 Data model issues that affect routing

The Router review found several details that can cause incorrect answers if the planner treats field names naively.

1. A sample call had `metadata.client = null`, while mentioned companies included organizations that were not the client. A company mention is therefore not a safe client key.
2. Presence flags such as pain points, objections and actions are booleans. They indicate existence, not item counts.
3. Final score belongs in metadata for numeric aggregation. Evaluation content is needed mainly for explanations.

4. Open questions can include answered items, so the collection name cannot be treated as equivalent to unanswered questions.
5. Uncertainties represent unresolved extraction or conversation ambiguity. They are not automatically client doubts about project qualification.
6. Evaluation scores and weights can be null. Aggregation must distinguish an unscored criterion from a scored zero.

These findings illustrate why a verified capability registry is stronger than a prompt full of guessed physical fields.

13.3 State, Memory and Conversational Continuity

13.3.1 Why state exists

A conversational query system should not force the user to repeat explicit scope on every turn. State allows a validated filter such as an SDR, call type or date range to persist while the user narrows the same exploration. Memory allows follow up language to be resolved from previous question and answer context.

13.3.2 Earlier state model

The earlier baseline used Session State and Session Memory. State persisted filters across turns, while memory preserved the previous exchange. A scope could be refined, replaced or reset. The architecture grouped related facets so switching from one consultant to another would remove conflicting consultant identifiers rather than intersecting two different people.

The presentation expresses the same concept through Query State, Last Messages State and Router Steps State. Query State represents the active scope, Last Messages State supports follow up detection and Router Steps State records current execution progress.

13.3.3 Current design direction

The newer minimal classifier already receives previous question and answer context before classification. The latest Router specification therefore avoids reintroducing hidden state fields into QueryIntent. It accepts only explicitly pinned context from the UI or pipeline and keeps the original question in the trace. This reduces stale filter risk while retaining conversational continuity where it is explicit.

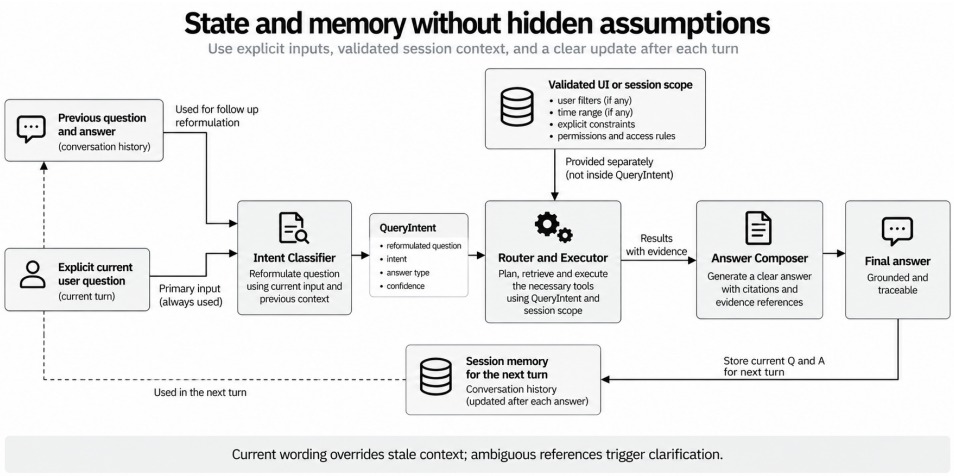


Figure 13.2: Conversation state diagram showing previous question and answer entering the classifier, explicit query scope entering the Router, and the final answer being written back to session memory

13.4 Answer Composition and Grounding

13.4.1 Answer Composer responsibility

The final model does not decide what to retrieve. It receives the user question, the expected answer shape and structured execution results. Its job is to synthesize only what the evidence supports and to communicate coverage gaps rather than inventing missing facts.

The earlier baseline defined answer confidence bands such as high, medium, low and insufficient data. Fallback data carried a lower confidence ceiling, and answers were expected to cite call identifiers and source evidence.

13.4.2 Why the final answer layer is separate

Separating answer generation from retrieval prevents the final model from silently changing the query. If the wrong calls were retrieved, we can inspect the Router and execution trace. If the right calls were retrieved but the wording is wrong, we can inspect the Answer Composer. This division of responsibility is important for debugging and for cost control.

13.4.3 Answer shape

The answer composer uses the classifier's `answer_type` to produce a suitable structure. A single value question should not receive a long report. A ranking should preserve order. A comparison should align the compared entities. A trend should preserve time order. A narrative may be longer, but it still must be grounded in retrieved evidence.

Chapter 14

Agent Validation, Demo and Maturity

14.1 What has been implemented

The project files support the existence of an earlier end to end baseline with modules for intent classification, routing, execution, pipeline state and memory, answer generation, ingestion and query APIs. The presentation also includes an agent demo and a slide dedicated to observed limitations.

The extraction pipeline itself has substantially stronger quantitative evidence because its models, prompt versions, chunk settings, schema versions and migration outputs were saved and benchmarked.

14.2 What has been validated in the latest Router design

The later four field Router specification was reviewed against ten difficult planning cases. The architecture document reports 13 ordered design sections, ten walkthroughs and twenty syntactically valid JSON examples. This is useful design validation because it checks contract consistency and edge cases. It is not the same as a live retrieval benchmark.

14.3 What we should not claim yet

The available latest Router document explicitly states that no live Elasticsearch query, classifier model call, production latency benchmark or executor integration test was performed for that design revision. Therefore we do not report an agent accuracy percentage or a latency winner for the latest architecture.

Table 14.1: Current evidence maturity by subsystem

Subsystem	Evidence status	Current interpretation
Real call audit	Confirmed	30 call manual evidence audit informs schema and rules.
Extraction model and segmentation	Confirmed quantitative	Multiple saved scorecards compare models, prompts, latency and cost.
Legacy versus Improved contract	Confirmed quantitative on two calls	Common information capture favors Improved on both evaluated cases.
Luna to Luna merge pair	Operating choice confirmed	A numerical pair benchmark is still missing from the identified evidence.
Earlier QA agent baseline	Implemented baseline confirmed	Code structure and demo oriented pipeline are documented.
Minimal four field classifier	Contract and prompt reviewed	Interface consistency issues were identified; broad classifier accuracy benchmark is not identified.
Latest hybrid Router	Design reviewed	Ten case architecture validation exists, but live integration benchmark remains pending.
GLM extension	Integration experiment	Structured output preflight succeeded after API adaptation; full comparable benchmark is incomplete.

14.4 Demo limitations observed

The presentation records four categories of demo risk: complex aggregation, precision on ambiguous questions, overly restrictive filters that return zero hits, and stale memory. The proposed mitigations include modular aggregation handling, systematic intent control before execution, progressive recall with validated adjustments, and strict session separation with explicit clearing policy.

The earlier baseline also revealed a deeper fallback risk. If a structured query returns zero, automatically escalating to broad semantic search can turn a correct zero into a weak positive answer. It can also hide extraction bugs by retrieving nearby uncertainty content instead of exposing that the expected field is empty. The roadmap therefore proposed an explicit policy for cases where zero is a legitimate answer.

Chapter 15

Challenges, Failures and Engineering Lessons

15.1 Identity and diarization

The most important identity lesson is that named attendance cannot be inferred from generic speakers or third person mentions. We addressed this with a conservative participant state and an explicit trusted SDR exception. The remaining risk is that trusted metadata itself can be wrong outside its authorized context, which is why the exception must remain visible and bounded.

15.2 Long context versus segmentation

Large context windows do not eliminate the need for segmentation experiments. Whole call input can be fast and cheap on short calls, but it can also miss distributed information. Very small chunks can improve local focus while increasing repeated prompt tokens, API calls, duplicate objects and cross chunk relationships. Our adaptive heuristic is a compromise rather than a theorem.

15.3 Cumulative state regression

Cumulative extraction appeared attractive because every new chunk could update one global state. In practice, later model calls could overwrite previously valid attributes, remove information or regress a lifecycle state. This led us to independent extraction and controlled reconciliation.

15.4 Schema growth versus semantic precision

Adding more fields created representational power, but it also created more boundaries the model could confuse. V7.1 therefore focused on field admission rather than simply adding fields. A technical difficulty is not automatically a client pain point. A suggestion is not automatically a need. A neutral price is not automatically an objection. This discipline is one of the central prompt engineering lessons of the project.

15.5 Evaluation fairness

The Legacy and Improved comparison initially risked being unfair because the extraction prompts influence what each schema is expected to contain. Rebuilding prompt aware references and then adding a shared common information ledger separated contract compliance from business information coverage.

15.6 Metrics can disagree

The short call Luna experiment showed that a composite audited quality score can improve while weighted field F1 remains similar or moves differently. This means a single aggregate score is not sufficient. We need at least structural validity, field coverage, unsupported additions, evidence validity, latency, request count and cost.

15.7 Model API compatibility

The GLM integration exposed a different class of issue. Two models may both support structured output but through different API interfaces. A benchmark harness must therefore abstract transport while recording the actual API path and schema mode. Otherwise we can mistake an API failure for a model quality failure.

15.8 Routing and data semantics

The agent work revealed that many apparent Router errors are actually data modeling ambiguities. A company mention is not a client identity. A boolean presence flag is not a count. An unanswered question is not equivalent to membership in a collection called open questions. Accurate Text to Query planning requires a capability registry that encodes these semantics.

Chapter 16

Roadmap and Remaining Validation

16.1 Router integration benchmark

The latest Router should then be evaluated end to end against a live Elasticsearch mapping. For every foundation question, the test should compare expected business operation, entity and date binding, physical query, returned call set, evidence coverage and final answer. Failures should be classified as classifier, entity resolution, capability mapping, plan validation, query compilation, data coverage or answer composition errors.

16.2 Retrieval improvements

The earlier architecture roadmap proposed embedding the semantic summary and combining structured retrieval with hybrid retrieval rather than relying only on a BM25 style cascade. A controlled retrieval benchmark should compare structured filtering, keyword semantic summary search and hybrid vector search using answerable call level questions. Any fallback must preserve the rule that a genuine zero result can be a correct answer.

16.3 Production observability

For every user request, we should store the raw question, exact QueryIntent, classifier prompt version, Router version, capability registry version, normalized dates, entity resolution result, operation shape, physical query in the executor trace, hit count, execution time, truncation status, returned call identifiers and evidence availability. This is enough to diagnose failures without exposing free form model reasoning.

Conclusion

The project began with a simple question: how can we make long DigiVoice conversations usable as reliable business data? The answer required more than choosing a language model. We first studied the calls, identified unreliable source assumptions, translated business questions into a structured information contract, tested model and prompt configurations, measured the effect of segmentation, rejected cumulative update behavior that could lose information, and retained independent extraction with controlled merge.

We then improved the extraction contract itself. The current design separates 16 business collections, uses conservative participant attendance, explicit project eligibility states, timestamp evidence, strict semantic admission gates and application derived metadata. The semantic summary is generated separately so it remains a retrieval representation rather than a source of truth.

The evaluation methodology also matured. We moved from native field scorecards to prompt aware references and a shared common information ledger. On the two evaluated migration calls, Improved captured more shared business information than Legacy, with gains of 10.0 and 18.3 percentage points. The reported cost increased by 0.13 dollars per 50 calls. These results justify the current direction for the evaluated cases, while the small sample requires a larger release benchmark before making a production wide quality claim.

Finally, we connected the extraction layer to a conversational question answering architecture. The earlier project baseline demonstrates an implemented end to end path with classification, routing, execution, state, memory and answer composition. The newer four field classifier and hybrid Router design improve responsibility boundaries and safety of planning, but this revision must still complete a live integration benchmark before it should be described as quantitatively validated.

The main engineering conclusion is therefore that reliability came from decomposition. We made the model responsible for semantic extraction, the merge responsible for reconciliation, application code responsible for deterministic metadata and evidence resolution, the Router responsible for planning, the Executor responsible for physical querying, and the Answer Composer responsible for grounded synthesis. This separation gives us a system that is easier to test, cheaper to query repeatedly and more transparent when it fails.

Bibliography

[1] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open Information Extraction from the Web," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

[2] L. Pai, W. Gao, W. Dong, L. Ai, Z. Gong, S. Huang, L. Zongsheng, E. Hoque, J. Hirschberg, and Y. Zhang, "A Survey on Open Information Extraction from Rule-based Model to Large Language Model," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9586–9608, 2024. doi: 10.18653/v1/2024.findings-emnlp.560.

[3] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural Architectures for Named Entity Recognition," in *Proceedings of NAACL-HLT*, pp. 260–270, 2016. doi: 10.18653/v1/N16-1030.

[4] G. Stanovsky, J. Michael, L. Zettlemoyer, and I. Dagan, "Supervised Open Information Extraction," in *Proceedings of NAACL-HLT*, pp. 885–895, 2018. doi: 10.18653/v1/N18-1081.

[5] L. Cui, F. Wei, and M. Zhou, "Neural Open Information Extraction," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 407–413, 2018. doi: 10.18653/v1/P18-2065.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019. doi: 10.18653/v1/N19-1423.

[8] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving Pre-training by Representing and Predicting Spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020. doi: 10.1162/tacl_a_00300.

[9] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi, "Entity, Relation, and Event Extraction with Contextualized Span Representations," in *Proceedings of EMNLP-IJCNLP*, pp. 5784–5789, 2019. doi: 10.18653/v1/D19-1585.

[10] Z. Zhong and D. Chen, "A Frustratingly Easy Approach for Entity and Relation Extraction," in *Proceedings of NAACL-HLT*, pp. 50–61, 2021. doi: 10.18653/v1/2021.naacl-main.5.

[11] G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. Anubhai, C. N. dos Santos, B. Xiang, and S. Soatto, "Structured Prediction as Translation between Augmented Natural Languages," in *International Conference on Learning Representations*, 2021.

[12] Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun, and H. Wu, "Unified Structure Generation for Universal Information Extraction," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 5755–5772, 2022. doi: 10.18653/v1/2022.acl-long.395.

[13] X. Wang, W. Zhou, C. Zu, H. Xia, T. Chen, Y. Zhang, R. Zheng, J. Ye, Q. Zhang, T. Gui, J. Kang, J. Yang, S. Li, and C. Du, "InstructUIE: Multi-task Instruction Tuning for Unified Information Extraction," *arXiv preprint arXiv:2304.08085*, 2023.

[14] Z. Wan, F. Cheng, Z. Mao, Q. Liu, H. Song, J. Li, and S. Kurohashi, "GPT-RE: In-context Learning for Relation Extraction using Large Language Models," in *Proceedings of EMNLP*, pp. 3534–3547, 2023. doi: 10.18653/v1/2023.emnlp-main.214.

[15] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, Y. Wang, and E. Chen, "Large Language Models for Generative Information Extraction: A Survey," *Frontiers of Computer Science*, vol. 18, no. 6, Art. 186357, 2024. doi: 10.1007/s11704-024-40555-y.

[16] Z. Zhang, W. You, T. Wu, X. Wang, J. Li, and M. Zhang, "A Survey of Generative Information Extraction," in *Proceedings of COLING 2025*, pp. 4840–4870, 2025.

[17] W. Zhang, W. Lu, J. Wang, Y. Wang, L. Chen, H. Jiang, J. Liu, and T. Ruan, "Unexpected Phenomenon: LLMs' Spurious Associations in Information Extraction," in *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9176–9190, 2024. doi: 10.18653/v1/2024.findings-acl.545.

[18] Y. Chen, Y. Liu, L. Chen, and Y. Zhang, "DialogSum: A Real-Life Scenario Dialogue Summarization Dataset," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 5062–5074, 2021. doi: 10.18653/v1/2021.findings-acl.449.

[19] M. Zhong, D. Yin, T. Yu, A. Zaidi, M. Mutuma, R. Jha, A. H. Awadallah, A. Celikyilmaz, Y. Liu, X. Qiu, and D. Radev, "QMSum: A New Benchmark for Query-based Multi-domain Meeting Summarization," in *Proceedings of NAACL-HLT*, pp. 5905–5921, 2021. doi: 10.18653/v1/2021.naacl-main.472.

[20] V. Rennard, G. Shang, J. Hunter, and M. Vazirgiannis, "Abstractive Meeting Summarization: A Survey," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 861–884, 2023. doi: 10.1162/tacl_a_00578.

[21] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[22] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big Bird: Transformers for Longer Sequences," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[23] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the Middle: How Language Models Use Long Contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024. doi: 10.1162/tacl_a_00638.

[24] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval," in *International Conference on Learning Representations*, 2024.

[25] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of EMNLP-IJCNLP*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

[26] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in *Proceedings of EMNLP*, pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

[27] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[28] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[29] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in *International Conference on Learning Representations*, 2023.

[30] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," in *International Conference on Learning Representations*, 2024.

[31] S. Es, J. James, L. Espinosa Anke, and S. Schockaert, "RAGAs: Automated Evaluation of Retrieval Augmented Generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pp. 150–158, 2024. doi: 10.18653/v1/2024.eacl-demo.16.

[32] Pydantic, *Pydantic Documentation: Models, Data Validation and JSON Schema*, official documentation, accessed September 2026.

[33] S. Ramírez and FastAPI Contributors, *FastAPI Documentation: Response Models, Validation and OpenAPI Integration*, official documentation, accessed September 2026.

[34] Elastic, *Elasticsearch Reference: Search, Query DSL, Mappings and Aggregations*, official documentation, accessed September 2026.

[35] Microsoft Research, *GraphRAG Documentation: Indexing, Querying and Knowledge Model Architecture*, official project documentation, accessed September 2026.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[37] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[38] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

[39] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of EMNLP-IJCNLP*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

[40] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv preprint arXiv:2312.10997*, 2023.

[41] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the Middle: How Language Models Use Long Contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024. doi: 10.1162/tacl_a_00638.

[42] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval," in *International Conference on Learning Representations*, 2024.

[43] Z. Li, C. Li, M. Zhang, Q. Mei, and M. Bendersky, "Retrieval Augmented Generation or Long-Context LLMs? A Comprehensive Study and Hybrid Approach," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 881–893, 2024. doi: 10.18653/v1/2024.emnlp-industry.66.

[44] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[45] Microsoft Research, *GraphRAG Documentation: Indexing Architecture, Knowledge Model and Query Engine*, official technical documentation, accessed September 2026.

[46] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in *International Conference on Learning Representations*, 2023.

[47] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," in *International Conference on Learning Representations*, 2024.

[48] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAS: Automated Evaluation of Retrieval Augmented Generation," *arXiv preprint arXiv:2309.15217*, 2023.

[49] D. Rau, H. Déjean, N. Chirkova, T. Formal, S. Wang, S. Clinchant, and V. Nikoulina, "BERGEN: A Benchmarking Library for Retrieval-Augmented Generation," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7640–7663, 2024. doi: 10.18653/v1/2024.findings-emnlp.449.

[50] X. Wang, Z. Wang, X. Gao, F. Zhang, Y. Wu, Z. Xu, T. Shi, Z. Wang, S. Li, Q. Qian, R. Yin, C. Lv, X. Zheng, and X. Huang, "Searching for Best Practices in Retrieval-Augmented Generation," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17716–17736, 2024. doi: 10.18653/v1/2024.emnlp-main.981.

[51] Pydantic, *Pydantic Documentation: Models, Validation, Serialization and JSON Schema*, official documentation, accessed September 2026.

[52] S. Ramírez and FastAPI contributors, *FastAPI Documentation: Tutorial, Data Validation, Response Models and API Development*, official documentation, accessed September 2026.

[53] Elastic, *Elasticsearch Reference: Indexing, Query DSL, Search, Aggregations and Mapping*, official documentation, accessed September 2026.

[54] Leyton, *CognitX and DigiVoice Internal Confluence Documentation*, internal company documentation, 2026. Restricted internal source.

[55] Leyton Data Science Team, *The Data Scientist Checklist*, internal technical reference, Leyton, 2026.

[56] Leyton CognitX, *SDR Call Pattern Analysis: 30 Calls*, internal DigiVoice audit report, 2026.

[57] Leyton CognitX, *DigiVoice Call Summary Extraction: Real-Data Findings, Benchmark Decisions and Schema Evolution*, internal technical project note, 7 September 2026.

[58] Leyton CognitX, *DigiVoice Extraction Pipeline Model, Prompt and Chunk Benchmark*, internal benchmark notebooks and experimental results, 2026.

[59] Leyton CognitX, *DigiVoice Structured Summary Schema and Extraction Prompt Specifications*, internal source code and prompt documentation, 2026.

[60] Leyton CognitX, *Prompt-Aware Legacy and Improved Ground Truth Evaluation*, internal evaluation notebooks for calls 6830 and 7517, 2026.

[61] Leyton CognitX, *DigiVoice Foundational Business Questions*, internal business requirements document, 2026.

Chapter A

Foundation Question Set

The following categories summarize the foundation questions used to shape the extraction and agent architecture.

A.1 Single call comprehension

1. What concerns or blockers did the client raise in this call?
2. What questions did the client ask the consultant?
3. What next steps were agreed, who owns them and by when?
4. Which projects were discussed and what was decided about each?
5. Who was on the call and on which side?
6. What documents were shared, requested or promised?

A.2 Single call quality

1. What was the score and the subscores for this call, with quoted evidence?
2. Where did this call lose points and what was the justification?
3. Did the consultant secure clear next steps with owners and timescales?

A.3 Consultant performance

Questions include average final score per consultant, highest and lowest consultants on a subcriterion, call volume per consultant, weakest team subcriterion, differences by call type and frequency of zero scores on next step ownership.

A.4 Time and volume

Questions include month to month score evolution, calls per week or month by call type, busiest periods and whether scores changed during those periods.

A.5 Content themes and client history

Questions include common client concerns, recurring information requests, calls with unresolved concerns and no agreed next step, complete history for one client company and change in eligibility doubts over time.

Chapter B

Current Prompt Contract in Report Form

This appendix documents the production prompt logic in a readable engineering form. It is not intended to replace the source `config.py`; the frozen source hash remains the authoritative reference when reproducing a benchmark.

B.1 Global chunk rules

```
Input: one transcript chunk plus trusted runtime metadata.
Output: one Summary matching the current Pydantic schema.

If the chunk has no substantive business content, return an empty Summary.
Missing in a chunk means not observed in this chunk, not absent from the call.
Do not reconstruct unseen earlier or later dialogue.
Overlap can repeat turns. Extract locally supported facts normally.
The merger owns duplicate reconciliation.
```

B.2 Evidence rules

```
Transcript evidence:
    source = TRANSCRIPT
    timestamp = exact source timestamp
    text = null
    speaker = null

Trusted configured SDR evidence:
    source = METADATA
    timestamp = null
    text = null
    speaker = null

Never fabricate, normalize, round, translate or reformat a transcript
   timestamp.
```

```
Do not copy transcript quotations into evidence text.
Post processing resolves readable evidence from the source record.
```

B.3 Participant rules

```
Trusted configured SDR:
    include once on the first chunk
    side = LEYTON
    role = SDR
    attendance = PRESENT

Every other named human from transcript:
    default attendance = MENTIONED_NOT_PRESENT
    upgrade the same human to PRESENT only with positive attendance evidence

Generic speaker labels are not names.
Role and side require explicit support, except the trusted SDR side.
Repeated mention does not prove presence.
```

B.4 Project eligibility rules

```
QUALIFIES          requires explicit final positive conclusion
LIKELY             requires explicit provisional positive conclusion
NEEDS_INFO         requires explicit statement that more information is needed
DOES_NOT_QUALIFY   requires explicit negative conclusion
NOT_DISCUSSED      means no project specific eligibility assessment appears

Never infer eligibility from complexity, innovation, experimentation or
   uncertainty.
```

B.5 Boundary rules

```
Isolated yes without visible proposition does not create an AGREED decision.
Solution proposal without client reaction keeps acceptance unknown.
Service explanation without client reaction does not create ServiceInterest.
Action suggestion without commitment is PROPOSED, not AGREED.
Question with answer outside the chunk remains unresolved locally.
Eligibility question without conclusion does not create positive eligibility.
```

B.6 Merge rules

```
Merge duplicates conservatively.
Preserve supported attributes, states and evidence.
Do not move one project's challenge, impact or evidence to another project.
Apply group eligibility only when the referenced project set is unambiguous.
Attach a later answer only to the question it clearly answers.
Resolve aliases only when identity equivalence is supported.
Preserve Uncertainty rather than guessing.
Do not perform fresh transcript extraction during merge.
```

B.7 Semantic summary rule

```
Run semantic summary generation after the final Summary is validated.
Use validated Summary information and trusted client context only.
Target one dense paragraph of approximately 70 to 100 words.
The semantic summary is a retrieval representation, not extraction truth.
```

Chapter C

Benchmark Interpretation Rules

C.1 Do not mix scorecards

The historical audited quality composite, weighted field F1, native prompt aware soft F1 and common information soft capture are different metrics. A model result from one scorecard should not be subtracted from a schema result in another scorecard and presented as one causal improvement.

C.2 Do not mix cost scopes

Historical benchmark costs are per tested call under the pricing assumptions saved in those notebooks. The later 0.87 and 1.00 dollar figures are project reported batch costs for 50 calls. These numbers must remain separate unless every included stage and price rate is aligned.

C.3 Distinguish model choice from contract choice

Luna and Terra are model choices. Legacy and Improved are schema plus prompt contract choices. Segmentation and merge model are additional dimensions. A statement such as “Improved is 14.2 points better” refers to the two common ledger call comparisons, not a direct Luna versus Terra result.