DeepSpecs: Expert-Level Question Answering in 5G

Aman Ganapathy Manvattira*, **Yifei Xu***, **Ziyue Dang, Songwu Lu**
University of California, Los Angeles
amanvatt02@g.ucla.edu, {yxu, ziyue.dang, slu}@cs.ucla.edu

Abstract

5G technology enables mobile Internet access for billions of users. Its design, implementation and operations are regulated by 3GPP standard specifications. We study *standard-native* question answering over 5G specifications, where expert-level queries require navigating thousands of pages of cross-referenced standards that evolve across tens of releases. Existing retrieval-augmented generation (RAG) frameworks, including telecom-specific approaches, rely on semantic similarity and cannot reliably resolve cross-references or reason about specification evolution. We present DEEPSPECS, a standard-native RAG system with three metadata-rich indices: SpecDB (clause-aligned specification text), ChangeDB (line-level version diffs), and TDocDB (Change Requests with design rationale). DEEPSPECS resolves cross-references by recursively retrieving referenced clauses via metadata lookup, and traces evolution by mining clause changes and linking them to corresponding Change Requests. We curate two 5G QA datasets: 573 expert-annotated real-world questions and 350 evolution-focused questions derived from approved Change Requests. Across multiple LLM backends, DEEPSPECS outperforms base models and state-of-the-art telecom RAG systems; ablations confirm that cross-reference resolution and evolution-aware retrieval substantially improve answer quality. Our methodology is conceptually applicable to other networked systems.

1 Introduction

5G technology enables mobile and wireless Internet access for billions of users globally. At the foundation of this ecosystem lies the 3GPP standards, a continuously evolving set of specifications that define the technical design and implementation of 5G systems. These documents span thousands of pages per release and cover diverse aspects such as architecture, protocols, operational procedures, and policy space. They serve as the operational reference for 5G mobile system practice (3rd Generation Partnership Project (3GPP), 2024a). For day-to-day operations, vendors and operators use concrete clauses to ensure interoperability across vendors and among components including devices, base stations, and core networks. They also rely on the standards to enforce conformance and certification, to fix field issues by tracing behaviors to the related clauses and their referenced conditions, and to schedule migrations to more recent releases by understanding what has been updated across releases and how those changes affect implementations. Similar workflows also appear in other networked systems whose standards evolve over time, such as Wi-Fi, Bluetooth, and Internet standards including DNS and BGP. Together, these practices highlight the need for *standard-native* question answering, where correctness depends on clause-level grounding, cited references, and updates across releases.

Recent advances in large language models (LLMs) and retrieval-augmented generation (RAG) have emerged as promising tools for automating technical question answering (Lewis et al., 2020; Guu et al., 2020; Izacard et al., 2023). Recent efforts have adapted these approaches to the telecom domain. Systems such as Tele-LLMs (Maatouk et al., 2024), Telco-RAG (Bornea et al., 2024), and Chat3GPP (Huang et al., 2025) incorporate domain-specific retrieval or fine-tuning on 3GPP specifications. Yet, they still largely follow vanilla RAG pipelines, which do not account for the requirements of standard-native usage. Consequently, two key challenges remain when applying these methods to 5G systems. *(1) Dense cross-references:* 5G specifications are designed for modularity and precision. Instead of repeating content, for readability they refer extensively to other clauses or

* Equal contribution.

Findings of the Association for Computational Linguistics: ACL 2026, pages 26935–26953
July 2-7, 2026

documents via specification and clause IDs (Qualcomm, 2017; 3rd Generation Partnership Project (3GPP), 2024b). As a result, critical definitions or conditions are often located outside of semantically similar text, requiring accurate reference resolution beyond what standard semantic retrieval can provide. *(2) Implicit reasoning behind the evolution of specifications:* As 5G specifications evolve incrementally over many releases, each version defines only what the intended behavior is at that point in time. Design rationale and historical context are excluded by intent, as specifications aim to specify "what is" rather than "why it changed." This makes it challenging to trace when and how a feature was introduced, modified, or deprecated without consulting external sources such as version changes and meeting discussions (Baron and Gupta, 2018; Chen et al., 2022).

To bridge these gaps, we design and implement **DEEPSPECS**, a 5G QA system tailored to the mobile Internet. DEEPSPECS extends the RAG paradigm with standard-native structural and temporal grounding over 3GPP specifications. Specifically, it performs standard-native indexing to build three metadata-rich indices from raw 3GPP corpora: *SpecDB* indexes clause-level specification content, *ChangeDB* tracks line-level edits across releases, and *TDocDB* organizes design-phase intents from 3GPP meeting documents (TDocs). Building on these indices, DEEPSPECS provides two clause-level capabilities that mimic how a real-world domain expert navigates the standards: (i) *cross-reference resolution* via rule-based reference extraction and hybrid retrieval, enabling fine-grained navigation within and across documents; and (ii) *specification-evolution reasoning* by mining clause changes and linking them to corresponding discussions through 3GPP-specific metadata, allowing the system to recall when a feature was introduced, how it evolved, and why.

To assess the effectiveness of our method, we curate a real-world dataset of 573 question-answer pairs spanning over 3 years from public telecom forums and blog posts, annotated by three telecom experts. The dataset covers practitioner-facing questions across RAN, core, protocol features, and evolution-related queries, providing a realistic testbed for telecom QA. DEEPSPECS demonstrates consistent gains over strong LLM baselines and a state-of-the-art telecom-specific RAG system. Ablations and microbenchmarks confirm the necessity and effectiveness of both cross-reference resolution and specification-evolution reasoning.

The code and datasets can be found on our website[1].

Contributions.

- We introduce DEEPSPECS, a framework for *standard-native* question answering in 5G that enables clause-level cross-reference resolution and specification-evolution reasoning, which are shared needs across standards-driven networked systems (e.g., Wi-Fi, Bluetooth, and Internet standards).
- We present, to our knowledge, the first expert-annotated 5G QA dataset, comprising 573 question-answer pairs collected from real-world practitioner forums and blog posts.
- Through automated and human evaluation, we show that DEEPSPECS outperforms strong LLM baselines combined with state-of-the-art retrieval, and ablate the contributions of cross-reference resolution and specification-evolution reasoning.

2 Related Work

Advanced RAG over Complex Corpora: Retrieval-augmented generation (RAG) systems such as Atlas (Izacard et al., 2023) and IRCoT (Trivedi et al., 2023) improve grounding by retrieving relevant context, but they are typically designed around isolated text chunks. For *standard-native QA* over technical specifications, this abstraction is often insufficient: answers frequently depend on resolving dense intra/inter-document cross-references and interpreting requirements as they evolve across releases, neither of which is explicitly modeled in standard RAG pipelines. Several research directions move toward handling more complex corpora. GraphRAG (Edge et al., 2024) represents text as knowledge graphs, agentic frameworks such as Legal Document RAG (Zhang et al., 2025a) and FinSage (Wang et al., 2025) use iterative or multi-stage retrieval and reasoning, and E^2RAG (Zhang et al., 2025b) explores retrieval over corpora with temporal characteristics. However, these approaches do not directly address the technical and versioned complexity of telecom standards, where correct answers often require precise reference resolution and release-aware interpretation grounded in clause-level normative

[1] http://metro.cs.ucla.edu/deepspecs.html

text. Meanwhile, more advanced general-purpose research and agentic frameworks (Jin et al., 2025; OpenAI, 2025; LangChain, 2025) emphasize multi-step exploration and synthesis but typically treat retrieval as a black-box primitive, further motivating standards-native indexing and retrieval as a complementary foundation.

Domain-Specific Telecom QA: LLMs have been applied to telecommunications to improve access to LTE/5G specifications. Tele-LLMs (Maatouk et al., 2024) train domain-specialized models on telecommunications corpora, while RAG-based systems such as Chat3GPP (Huang et al., 2025), Telco-RAG (Bornea et al., 2024), and Telco-oRAG (Bornea et al., 2025) combine hybrid retrieval and indexing choices for QA over 3GPP corpora, with query enhancement and routing components emphasized in some systems. Despite these advances, most systems still retrieve primarily by semantic similarity over text chunks, which makes it hard to reliably traverse dense cross-references or answer evolution-centered questions that require tracing changes across releases. Complementary resources such as TSpec-LLM (Nikbakht et al., 2024) provide broad 3GPP coverage, but focus on multiple-choice evaluation and do not target reference resolution or release-aware reasoning. These gaps motivate standards-native indexing and retrieval for expert-level telecom QA.

3 Method

In this work, we focus on the topic of QA (question and answering) over 3GPP 5G specifications where the information is *standard native*: correct answers depend on how the standard is written and maintained, not only on surface semantics. In particular, human domain experts routinely (i) follow explicit citations across specifications and clauses to reconstruct the necessary context chain, and (ii) reason over release history to determine when a behavior was introduced, how it was revised, and what motivated the change. Our goal is to build a system that mirrors this practice of top system experts by producing answers that are both accurate and explanatory, grounded in cited clauses and the design rationale recorded during the standardization process.

Our problem can be illustrated through the following QA pair from a real-world telecom forum:

> **Q:** UE supports VoLTE and SA, but not VoNR ... When a call is triggered in 5G SA, and if the network configures the VoNR bearer (ignoring UE capability) what should the UE do here? Any specification reference would be helpful.
> **A:** The UE should reject the VoNR bearer establishment attempt with an appropriate cause code. According to 3GPP **TS 24.501 (5G NAS protocol), section 6.4.1.3** ... Additionally, 3GPP **TS 38.331 (5G RRC protocol)** specifies ... For fallback procedures, 3GPP **TS 23.502 section 4.13** details ... The UE should trigger EPS fallback to use its VoLTE capability instead.

The above example shows the main challenge of standard-native QA: the needed components are scattered across multiple standards of TS 24.501, TS 38.331, and TS 23.502. Determining which subclauses and procedures to follow (e.g., rejection versus fallback) typically depends on how capability handling and fallback behavior were introduced and updated across releases, with the underlying rationale recorded in 3GPP meeting archive (e.g., request for changes) rather than stated explicitly in the specifications.

3.1 Overview

Our proposed DEEPSPECS answers expert-level questions over 5G specifications by extending a RAG framework with structural and temporal reasoning. Emulating human expert practice, it supports clause-level *cross-reference resolution* and *specification-evolution reasoning*.

At the high level, DEEPSPECS operates in four stages: *(1) Standard-native indexing*, where specification content and TDocs are indexed into three retrieval indices (Figure 1); *(2) Cross-reference resolution*, which retrieves cited clauses using metadata and recursively follows references (Figure 2, upper); *(3) Specification-evolution reasoning*, which traces relevant changes across releases and retrieves supporting design-phase rationale from TDocs (Figure 2, lower); and *(4) Answer generation*, which synthesizes the final response grounded in the retrieved evidence.

3.2 Standard-Native Indexing

We index 5G specifications in the way that preserves two properties that dominate expert usage of the standards: modularity across clauses and evolution across releases. This allows downstream components to operate over three standard-native indices that align with the anchors practitioners use (Figure 1).

Clause as the unit of indexing. We treat an atomic clause (e.g., "7.4.1.1.2 Mapping to physical resource") as the basic indexed unit, matching

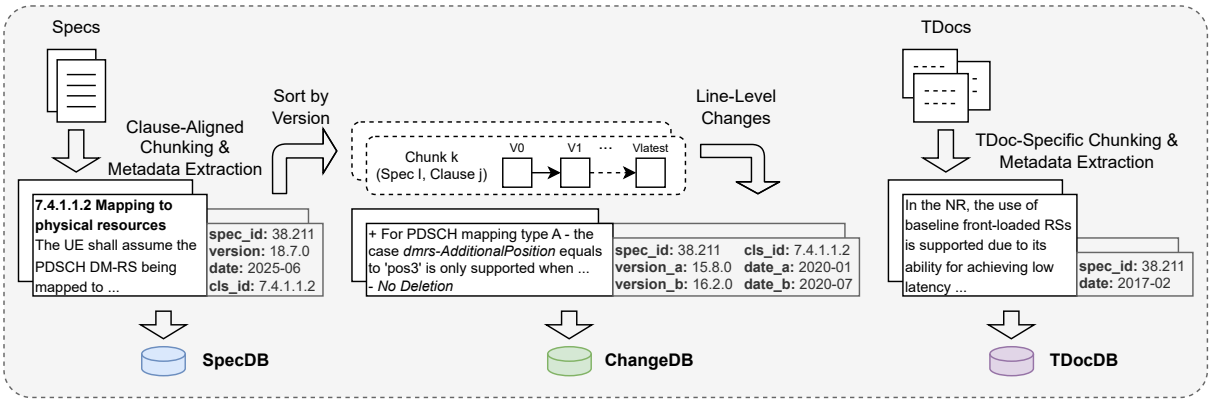

Figure 1: Standard-native indexing in DEEPSPECS. The system builds three retrieval indices from 5G specifications and TDocs, each enriched with structured metadata: SpecDB indexes clause-level specification content; ChangeDB captures clause-level edits across releases; TDocDB indexes change requests to provide design-phase rationale.

how 3GPP text is cited and discussed. From official DOCX specifications in the 3GPP archive[2], we segment documents into clause-aligned chunks and attach structured identifiers and version information, including specification ID, clause ID, version number, and timestamp, extracted from consistent document patterns (OpenAirInterface, 2022) with an LLM-based extractor (Appendix A.1). These clause chunks form *SpecDB*, our primary index over specification content.

Indexing revision history. To make evolution queries tractable, we align clause snapshots across versions using the shared `<spec_id, cls_id>` key and order them by timestamp. We then compute line-level diffs between adjacent observed versions and index each edit as a standalone record with before-and-after metadata. This produces *ChangeDB*, which represents what changed and when, and treats the earliest observed snapshot of a clause as its initial introduction. Below is a partially omitted sample chunk from ChangeDB.

> **[TS 38.331, v18.1.0->18.2.0, Clause 5.3.5.8.2, addition]** NOTE 4: It is up to UE implementation whether. . .

Indexing design-phase context. While specifications describe the final agreed behavior, the design-phase motivations and trade-offs are typically recorded in 3GPP meeting documents (TDocs) (ShareTechnote; Baron and Gupta, 2018). In this work, we focus on *Change Requests* (CRs), which are tightly coupled to specification edits and follow a consistent structure in the archive. We collect DOCX-formatted CRs and build *TDocDB* with CR-specific chunking and metadata extraction. We parse three sections with an LLM-based extractor (Appendix A.2): the summary of changes, the reasons for the change, and the consequences of non-approval. Each individual change listed in the summary is segmented into a separate chunk and paired with its corresponding rationale and consequences. Below is a sample chunk from TDocDB.

> **[TS 38.213, v17.1.0, 2022-05]**
>
> **Change Summary:** Clarify that a CSI-RS is an aperiodic CSI-RS resource in a CSI-RS resource set in clause 6.
>
> **Reason:** Unclear association of CSI-RS that shares the same indicated TCI state as for the PDCCH and PDSCH in clause 6.
>
> **Consequence if this change is not accepted:** Incomplete support for MIMO enhancements in NR.

All three indices store chunk text together with structured metadata enabling both exact filtering and dense representations for semantic ranking.

3.3 Cross-Reference Resolution

The 5G technical specifications are intentionally modular. To reduce redundancy and maintain formality, clauses frequently cite other clauses within the same document or across documents. For instance, a clause in TS 38.211 may define a feature but defer key operational conditions to TS 38.214. As a result, answering a single technical query often requires assembling a clause chain by following these citations (OpenAirInterface, 2022).

DEEPSPECS operationalizes this practice by resolving cross-references at the clause level (Figure 2, upper). We first retrieve the top-k_1 candidate clauses from SpecDB via dense retrieval; we use HyDE (Gao et al., 2023) to mitigate the common

[2] https://www.3gpp.org/ftp/

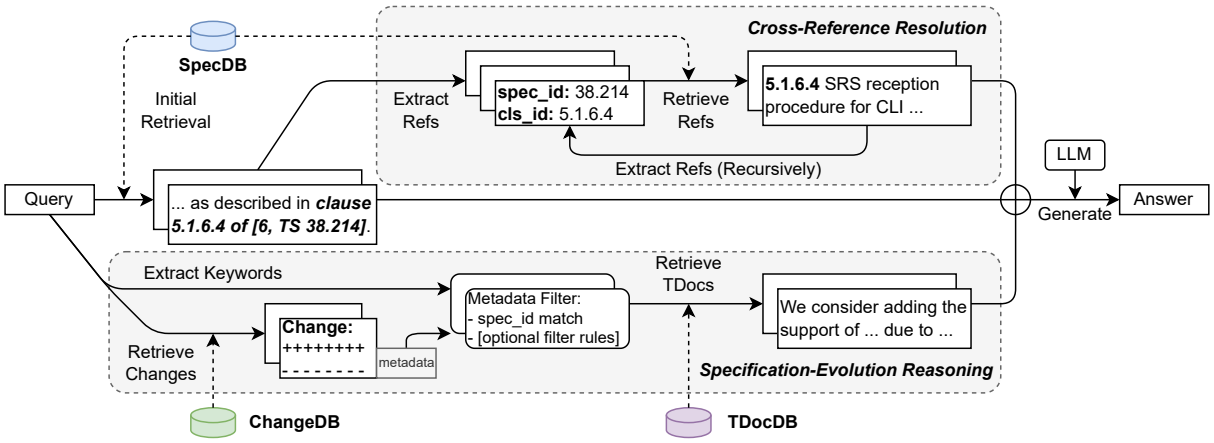

Figure 2: Retrieval in DEEPSPECS. The system leverages structured metadata in the three indices to resolve cross-references (upper) and trace specification evolution with supporting CR rationale (lower), enabling standard-native question answering with structural and temporal grounding in 5G specifications.

case where real-world questions are underspecified and sparse in semantic cues, improving zero-shot retrieval. From the retrieved clauses, we extract cited <spec_id, cls_id> pairs using regular expressions that match common 3GPP citation patterns (e.g., "See clause 5.1.6.4 of TS 38.214."), with the full set of patterns listed in Appendix B. We then resolve each extracted pair through metadata lookup in SpecDB to obtain the referenced clause text and recursively expand newly retrieved clauses up to a maximum depth to recover multi-hop dependencies. Finally, we re-rank all collected clauses by semantic relevance to the original query and return the top-k_2 clauses for answer generation.

3.4 Specification-Evolution Reasoning

Evolution-oriented questions require reasoning over *unique relations* across documents and versions: the relevant evidence is not a single clause, but the link between a specific edit (what changed and when) and the design intent recorded around it. Rather than retrieving TDocs purely by semantic similarity, DEEPSPECS uses ChangeDB as an explicit bridge from the user question to the most relevant CR rationale, yielding a traceable path from a concrete revision to its supporting discussion (Figure 2, lower).

Given a query, DEEPSPECS routes it through the most reliable anchor available. If the user mentions a specification explicitly, we extract the spec ID with an LLM extractor (Appendix A.3) and use it to constrain the candidate set. Otherwise, we retrieve candidate change entries from ChangeDB using dense retrieval with HyDE, which is robust to underspecified, identifier-heavy questions and directly surfaces additions, removals, or modifications. We then use the metadata of the selected change entry (e.g., spec ID and timestamp) to filter candidate CRs in TDocDB; our current implementation filters by spec ID, which is sufficiently accurate for CRs, and the framework supports additional metadata constraints when other TDoc types are included. Finally, we re-rank the filtered CR chunks by relevance to the HyDE-generated hypothesis for the question and return the top-k_3 rationale chunks for answer generation.

This hybrid linkage grounds evolution reasoning in an explicit change record, improving interpretability compared to purely semantic TDoc retrieval while providing the explanatory context practitioners often seek.

We explore the retrieval process in specification-evolution reasoning with an illustrative example in Appendix I.

3.5 Answer Generation

After retrieving relevant specification clauses, resolving cross-references, and identifying supporting TDocs, DEEPSPECS assembles a generation prompt for a general-purpose language model to synthesize the final answer. It includes the top-k_1 clauses from initial retrieval, the top-k_2 clauses from cross-reference resolution, and the top-k_3 TDoc chunks from specification-evolution reasoning, each selected by semantic similarity. The complete prompt template is provided in Appendix A.4.

3.6 An End-to-End Case Study

We walk through DEEPSPECS's full retrieval process with an example in Appendix J. We describe

how cross-reference resolution and specification-evolution reasoning can empower more targeted answer generation.

4 Datasets

There are very few benchmarks established for telecom QA and most emphasize surface-level factual queries, rely on synthetic QA collections, and often restrict evaluation to multiple-choice problems (Maatouk et al., 2024; Huang et al., 2025; Bornea et al., 2024, 2025; Nikbakht et al., 2024). We address these gaps by constructing two new datasets that capture (i) real-world practitioner information needs and (ii) reasoning behind the evolutions of specifications.

4.1 Real-World QA Dataset

To evaluate system performance in realistic professional settings, we curate a dataset of question–answer pairs that reflect genuine practitioner information needs. The data are drawn from publicly available forums and educational resources where telecommunications professionals actively exchange knowledge on 5G, ensuring alignment with engineering and instructional use cases.

Data Collection. We collect QA data from two widely used sources spanning a three-year period from August 2022 to August 2025. *telecomHall* (telecomHall) is a long-standing global telecommunications community (established in 1999) where practitioners discuss technical challenges. We retain only explicit question–answer exchanges that are technically accurate, complete, and well-formed. *ShareTechnote* (Ryu) is an educational resource with extensive technical documentation on cellular communications, with a strong emphasis on 5G. From 236 webpages, we formulate precise questions and extract answers directly from the material. The questions are categorized into five technical domains (detailed taxonomy in Appendix E).

Human Annotation. We recruit three volunteers with advanced backgrounds in mobile networks and cellular systems to collect and validate the QA pairs. Recruitment protocols and detailed annotation guidelines are provided in Appendix C.

4.2 CR-Focused QA Dataset

To support targeted evaluation of DEEPSPECS specification-evolution reasoning, we construct another QA dataset of derived from real 3GPP specification changes with rich supervision signals.

Source	#	Avg. Q Len	Avg. A Len
Real-world QA Dataset			
telecomHall	102	46.9 tokens	126.9 tokens
ShareTechnote	471	30.3 tokens	102.7 tokens
Total	573	33.2 tokens	107.0 tokens
CR-focused QA Dataset			
3GPP Change Requests	350	45.4 tokens	287.7 tokens

Table 1: Statistics of the QA datasets, including real-world sources and CR-based pairs.

Data Collection. We collect approved Change Requests from 3GPP specifications spanning Releases 17 and 18. The collection yields 997 CR documents. Each approved CR is retrieved with its associated TSG documentation package containing the complete change proposal, rationale, technical details, and impact analysis.

QA Generation. We use a combination of LLM and human verification to generate the QA pairs. We first extract three key fields from each CR: summary of change, reason for change, and consequences if not approved. To ensure quality, we filter out trivial CRs where only minor editorial corrections are made. For substantive CRs, we employ an LLM to generate question-answer pairs following a structured prompt (detailed in Appendix A.7). All generated QA pairs are manually reviewed to verify technical accuracy, question practicality, and answer quality.

4.3 Dataset Characteristics.

Table 1 summarizes statistics of the final collected real-world and CR-focused QA datasets. All sources are publicly available. Our use of these sources for research and system evaluation purposes aligns with the educational and knowledge-sharing intent of these platforms, and our created artifacts are intended solely for academic research and development of QA systems in the telecom domain.

5 Experiments

5.1 Experimental Setup

5.1.1 Retrieval Corpora

We collect all Release 17 and 18 3GPP technical specifications, following prior work (Huang et al., 2025). In total, we obtain 2137 specifications

as input for database construction. We download Change Requests from 3GPP Releases 17 and 18 that satisfy "approved" status requirements, yielding 997 TDocs in total.

5.1.2 Baselines

We compare DEEPSPECS against three categories of baselines: **(1) Base Model:** Direct prompting of the base LLMs without any retrieval context. For consistency, we use the same answer-generation prompt (Appendix A.4) but replace the context section with "`NO CONTEXT`." **(2) Chat3GPP:** A state-of-the-art RAG system tailored for telecom-domain question answering (Huang et al., 2025) employs hybrid retrieval, specialized chunking, and efficient indexing, and demonstrates superior performance over existing methods. Nevertheless, Chat3GPP still largely follows the vanilla RAG paradigm. **(3) Telco-oRAG:** An advanced RAG framework tailored for telecommunications queries (Bornea et al., 2025) introduces glossary-enhanced query refinement, dual-round hybrid retrieval combining 3GPP databases with real-time web search, and a memory-efficient neural router. While it extends Telco-RAG (Bornea et al., 2024) with web search, it does not address structural or temporal reasoning. We use Chat3GPP as our primary baseline because it is the state-of-the-art telecom RAG system and has reported gains over Telco-RAG on TeleQnA. Telco-oRAG extends Telco-RAG with web search but does not explicitly address structural or temporal reasoning. Comparisons against Telco-oRAG have been deferred to Appendix D.

For all baselines, we test a collection of generation LLMs with their default or recommended hyperparameter settings.

5.1.3 Evaluation Metrics

Pairwise Win Rate. We employ an LLM evaluator validated with human evaluation to compute head-to-head win rates between system outputs. The evaluator is given the question, a gold answer, and two candidate answers, and is instructed to select the preferred one based on accuracy and helpfulness. The evaluation prompt is provided in Appendix A.5 and the model used is GPT-4o. To mitigate positional bias, evaluation is repeated with reversed answer order. The judge is explicitly grounded by a reference answer. During evaluation, we provide: (i) the question, (ii) the reference answer (which the generators never see), and (iii) two candidate answers. The judge is instructed to select the answer that is more accurate and helpful with respect to the reference. This setup reduces reliance on the judge's internal domain knowledge because it is comparing candidates against an external anchor, rather than generating answers from scratch. We validate the alignment with human evaluators by asking domain experts to re-judge a subset of 144 pairs. The LLM-based judgments are aligned with human assessments in 92.4% of cases, with most disagreements occurring in close comparisons.

Rubric-Based Scoring. To complement pairwise win rates with absolute quality scores, we design question-specific rubrics. For each QA pair, rubrics are first generated with an LLM, then curated and validated by domain experts. During evaluation, an LLM applies these rubrics to score each candidate answer along multiple dimensions. The rubric design and evaluation prompts are provided in Appendix A.6.

We elect not to use similarity based metrics like ROUGE or BERTScore. Even with gold answers, in our open-ended QA task, answer quality cannot be reduced to binary correctness due to the relevance of other factors such as explanation helpfulness. Additionally, such metrics are shown to correlate poorly with answer quality in such settings (Wang et al., 2023; wai Yim et al., 2025). In the following experiments, we use the Pairwise Win Rate as the primary metric as it has been shown to be more reliable and to align better with human judgment (Liusie et al., 2024; Liu et al., 2024). We have deferred additional results for rubric-based scoring to Appendices E.3 and F.

We provide full implementation details for our experiments in Appendix H.

5.2 Results on Real-World QA

Table 2 summarizes results across multiple model families. For DEEPSPECS, we fix retrieval parameters (k_1=4, k_2=3, k_3=3). We configure Chat3GPP to return the top-10 chunks so that DEEPSPECS and Chat3GPP use comparable context budgets. We report both pairwise win rates and output lengths to control for verbosity effects.

DEEPSPECS yields consistent gains over strong baselines. Retrieval augmentation improves performance across all backends: both Chat3GPP and DEEPSPECS outperform their corresponding base models, highlighting the importance of grounding for navigating 3GPP specifications. However,

Table 2: Comparative performance on the real-world QA dataset. Both Chat3GPP and DEEPSPECS return the top-10 chunks. All pairwise comparisons are made against a GPT-4o baseline. Note that the win rate for GPT-4o is calculated from a self-to-self comparison and therefore always 50%.

Model	Method	Length (tokens)	Win Rate vs. GPT-4o (%)
GPT-4o	Base Model	295	(50)
	Chat3GPP	295	62.5
	DEEPSPECS	301	**68.1**
Qwen3-4B	Base Model	307	34.4
	Chat3GPP	427	67.3
	DEEPSPECS	415	**71.2**
Qwen3-8B	Base Model	346	50.8
	Chat3GPP	490	74.4
	DEEPSPECS	483	**79.8**
Qwen3-14B	Base Model	331	64.7
	Chat3GPP	459	78.9
	DEEPSPECS	460	**83.1**
Qwen3-32B	Base Model	373	73.6
	Chat3GPP	499	82.9
	DEEPSPECS	497	**88.0**
Claude 3.5 Haiku	Base Model	268	48.6
	Chat3GPP	278	56.6
	DEEPSPECS	278	**59.4**
GPT-4.1 mini	Base Model	395	82.0
	Chat3GPP	424	87.8
	DEEPSPECS	442	**90.1**
GPT-4.1	Base Model	427	87.2
	Chat3GPP	494	93.7
	DEEPSPECS	511	**95.4**

Chat3GPP primarily relies on semantic similarity retrieval, which limits its ability to systematically follow cross-references or reason about release evolution. DEEPSPECS addresses these gaps with structural- and evolution-aware retrieval, producing consistent improvements across generator models and question categories, without relying on longer outputs. As with any retrieval system, the initial step may introduce irrelevant content. However, the generator ultimately decides how to use the retrieved context so that in the worst case DEEPSPECS behaves similarly to Chat3GPP.

Effect of model capacity and thinking mode. Gains are larger for smaller or weaker backends but remain meaningful for stronger models, indicating that even high-capacity LLMs benefit from explicit grounding in clause structure and temporal evolution of the standards. For Qwen backends, we enable thinking mode unless specified; Appendix E.2 compares thinking vs. non-thinking settings and shows that thinking mode generally helps, while DEEPSPECS remains beneficial regardless of modes.

Table 3: Ablation studies performed on the real-world QA dataset. Both Chat3GPP and DEEPSPECS return the top-10 chunks.

Model	Method	Length (tokens)	Win Rate vs. GPT-4o (%)
GPT-4o	Base Model	295	(50)
	Chat3GPP	295	62.5
	DEEPSPECS (w/o cross-ref)	297	**64.6**
	DEEPSPECS (w/o spec-evol)	299	**66.5**
	DEEPSPECS (full)	301	**68.1**
Qwen3-8B	Base Model	346	50.8
	Chat3GPP	490	74.4
	DEEPSPECS (w/o cross-ref)	482	**77.8**
	DEEPSPECS (w/o spec-evol)	506	**77.7**
	DEEPSPECS (full)	483	**79.8**
Qwen3-32B	Base Model	373	73.6
	Chat3GPP	499	82.9
	DEEPSPECS (w/o cross-ref)	498	**86.6**
	DEEPSPECS (w/o spec-evol)	489	**86.8**
	DEEPSPECS (full)	497	**88.0**

Rubric-based evaluation corroborates pairwise results. Rubric scoring confirms these trends. With GPT-4.1, DEEPSPECS improves the mean score by 0.62 on a 5-point scale (Wilcoxon signed-rank test: $p < 0.0001$, d_z=0.41), and with GPT-4.1-mini by 0.60 ($p < 0.0001$, d_z=0.40). Per-category win rates and rubric details are reported in Appendix E.

5.3 Ablation Studies

To better demonstrate the impacts of the system components, we perform ablation studies on the real-world QA dataset. We show the impact of spec-evolution reasoning by disabling cross-reference resolution, with the parameters set to (k_1=7, k_2=0, k_3=3). Similarly, we measure the impact of cross-reference resolution by disabling spec-evolution reasoning and setting the parameters to (k_1=7, k_2=3, k_3=0). The results, seen in Table 3, validate each component's improvement over Chat3GPP.

Where labels are available, Our microbenchmark in Appendix G further quantifies our retrieval performance, showing that DEEPSPECS substantially improves F1 over Chat3GPP therefore validating the misalignment between semantic retrieval and cross-reference resolution.

5.4 Results on CR-Focused QA

We isolate the contribution of evolution reasoning using a CR-focused QA dataset. To control for confounders, we disable reference resolution and fix retrieval parameters (k_1=5, k_2=0, k_3=5), ensuring DEEPSPECS and Chat3GPP retrieve the same

Table 4: Comparative performance on the CR-Focused QA Dataset. Even with reference resolution disabled, DEEPSPECS outperforms the base model and Chat3GPP at comparable output lengths.

Model	Method	Length (tokens)	Win Rate vs. GPT-4o (%)
GPT-4o	Base Model	273	(50)
	Chat3GPP	265	59.3
	DEEPSPECS	269	**77.0**
Qwen3-8B	Base Model	312	73.3
	Chat3GPP	466	85.7
	DEEPSPECS	427	**92.6**
Qwen3-32B	Base Model	353	89.9
	Chat3GPP	549	93.7
	DEEPSPECS	506	**95.7**

number of chunks. Table 4 reports GPT-4o results. Relative to both the base model and Chat3GPP, DEEPSPECS (with reference resolution disabled) still achieves significantly higher win rates while producing outputs of comparable length. Results for additional models are provided in Appendix F and show the same trend. Rubric-based evaluation reinforces these findings. On this CR-focused subset, DEEPSPECS achieves a mean quality score of 3.05, compared to 2.48 for Chat3GPP and 2.26 for the base model (all differences significant at $p<0.0001$, detailed scores in Appendix F).

6 Conclusion

We introduce DEEPSPECS, a standard-native QA framework that incorporates the structural and temporal properties of 3GPP specifications. DEEPSPECS performs clause-level cross-reference resolution and specification-evolution reasoning, capabilities largely absent from standard RAG yet critical for grounded and helpful 5G question answering. Evaluations on real-world questions and evolution-focused benchmarks show consistent gains over strong LLM and telecom RAG baselines, with both structural and temporal components contributing to performance.

Limitations

Extensibility. This work focuses on the 5G mobile Internet and its 3GPP corpus, where specifications exhibit dense cross-references and frequent release-to-release evolution. These properties also hold for other networked sytems, including Wi-Fi technology family (from 802.11a/b/g/n to Wi-Fi 6 and 802.11d), Internet subsystems (e.g., DNS and BGP), Bluetooth, Zigbee, etc.. Our proposed methodology is also conceptually applicable to such systems. However, transferring our approach to these domains is not automatic. For Wi-Fi, this would require constructing a comparable corpus across IEEE 802.11 standards across different versions and organizing vendor feature documentation so that version- and feature-specific behaviors can be traced to the appropriate specification. For Internet standards such as DNS and BGP, it would require curating relevant RFCs and updates, resolving cross-document references, and aligning changes across revisions and errata. In summary, our methodology is applicable in principle, but requires domain-specific corpus construction, reference parsing, and the availability of revision and discussion artifacts. We view these domains as promising next steps and are actively exploring them. The systematic cross-domain evaluation is beyond the scope of this paper.

Coverage of Specifications and TDocs. Our study focuses on English NR (3GPP) specifications, primarily Releases 17–18, and question types skewed toward RAN/PHY procedures. For TDocs, we currently include only CR-type documents. We have not evaluated on other TDoc types, earlier or future releases, or multilingual corpora, so our claims should not be over-generalized.

Structural Assumptions on Documents. The method assumes consistent clause numbering, machine-parseable cross-references, and semi-templated Change Requests (CRs). While this holds for most 3GPP documents, exceptions do occur (e.g., atypical numbering, unusual references, or editorial inconsistencies). Such cases can reduce the reliability of linking and reasoning across documents.

Dependence on LLM-Based Processing. Our pipeline makes extensive use of large language models for chunking, extraction, and metadata handling. This introduces potential errors and computational or API calling cost. However, most of the processing is one-time or incremental with new releases, and we found that budget-oriented models are sufficient for these tasks. Still, downstream accuracy ultimately depends on model quality, which may vary with version updates.

Potential of Inaccurate Retrieval Although DEEPSPECS employs deterministic rule-based extraction to minimize unexpected retrievals, pipeline

errors can still occur. Imperfect metadata extraction or inaccurate semantic hits can occasionally propagate irrelevant cross-references or mismatched temporal rationales into the final prompt. However, we observe that the LLM generator acts as a robust natural guardrail, frequently ignoring noisy or irrelevant evidence during answer synthesis. Empirically, this error propagation does not dominate overall performance: DEEPSPECS consistently outperforms baselines across all LLM backends as seen in Table 2, indicating that the strong signal from accurate structural and temporal reasoning strongly outweighs occasional retrieval noise. Finally, DEEPSPECS exposes configurable retrieval budgets (k_1, k_2, k_3) and recursion depths, allowing practitioners to explicitly manage the precision-recall trade-off in deployment.

Evaluation. (1) Our annotated datasets are limited in size due to the need for careful expert curation and limited number of high-quality data sources. This constraint may affect statistical significance, and performance estimates should be interpreted with caution. (2) For our evaluation, some evaluation metrics rely on LLM-based judges or scorers. Although we include human validation, LLM evaluators can reward fluency or plausible reasoning over our desired factors, and prompt bias may persist. We view LLM-based evaluation as a useful complement rather than a full substitute for human evaluation.

Potential Risks. Our system generates fluent, citation-backed outputs, but answers may still be incomplete or misleading if context is missed. Over-reliance on such outputs without cross-checking against the specifications could encourage misinterpretation or non-compliant implementations. As our work is based on public 3GPP standards, direct risks related to privacy, security, or fairness are minimal. We encourage deployment with safeguards such as citation enforcement, uncertainty reporting, and access controls.

References

3rd Generation Partnership Project (3GPP). 2024a. 5g system overview. Accessed: 2025-07-28.

3rd Generation Partnership Project (3GPP). 2024b. Specifications and technologies. Accessed: 2025-07-28.

Anthropic. 2024. Claude 3.5 haiku. https://www.anthropic.com/claude/haiku. Accessed: 2025-10-06.

Justus Baron and Kirti Gupta. 2018. Unpacking 3gpp standards. *Journal of Economics & Management Strategy*, 27(3):433–461.

Andrei-Laurentiu Bornea, Fadhel Ayed, Antonio De Domenico, Nicola Piovesan, and Ali Maatouk. 2024. Telco-rag: Navigating the challenges of retrieval augmented language models for telecommunications. In *GLOBECOM 2024-2024 IEEE Global Communications Conference*, pages 2359–2364. IEEE.

Andrei-Laurentiu Bornea, Fadhel Ayed, Antonio De Domenico, Nicola Piovesan, Tareq Si Salem, and Ali Maatouk. 2025. Telco-orag: Optimizing retrieval-augmented generation for telecom queries via hybrid retrieval and neural routing. *arXiv preprint arXiv:2505.11856*.

Yi Chen, Di Tang, Yepeng Yao, Mingming Zha, XiaoFeng Wang, Xiaozhong Liu, Haixu Tang, and Dongfang Zhao. 2022. Seeing the forest for the trees: Understanding security hazards in the {3GPP} ecosystem through intelligent analysis on change requests. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 17–34.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Long Huang, Ming Zhao, Limin Xiao, Xiujun Zhang, and Jungang Hu. 2025. Chat3gpp: An open-source retrieval-augmented generation framework for 3gpp documents. In *2025 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 492–497.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

LangChain. 2025. Open deep research. https://github.com/langchain-ai/open_deep_research. Accessed: 2026-01-04.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2024. Aligning with human judgement: The role of pairwise preference in large language model evaluators. *arXiv preprint arXiv:2403.16950*.

Adian Liusie, Potsawee Manakul, and Mark Gales. 2024. Llm comparative assessment: Zero-shot nlg evaluation through pairwise comparisons using large language models. In *Proceedings of the 18th conference of the European chapter of the Association for Computational Linguistics (volume 1: long papers)*, pages 139–151.

Ali Maatouk, Kenny Chirino Ampudia, Rex Ying, and Leandros Tassiulas. 2024. Tele-llms: A series of specialized large language models for telecommunications. *arXiv preprint arXiv:2409.05314*.

Rasoul Nikbakht, Mohamed Benzaghta, and Giovanni Geraci. 2024. Tspec-llm: An open-source dataset for llm understanding of 3gpp specifications. *arXiv preprint arXiv:2406.01768*.

OpenAI. 2024. Gpt-4o system card. https://openai.com/index/gpt-4o-system-card/. Accessed: 2025-10-06.

OpenAI. 2025. Gpt-4.1 (and variants: mini, nano). https://openai.com/index/gpt-4-1/. Accessed: 2025-10-06.

OpenAI. 2025. Introducing deep research. https://openai.com/index/introducing-deep-research/. Accessed: 2026-01-04.

OpenAirInterface. 2022. Walkthrough 3gpp specifications. Accessed: 2025-07-28.

Qualcomm. 2017. Understanding 3gpp: Starting with the basics. Accessed: 2025-07-28.

Jaeku Ryu. Sharetechnote. https://www.sharetechnote.com. Accessed: 2025-10-04.

ShareTechnote. 5g frame structure and related discussions: Background context and rationale. Accessed: 2025-07-28.

telecomHall. telecomhall forum. https://www.telecomhall.net. Accessed: 2025-10-04.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037.

Wen wai Yim, Asma Ben Abacha, Zixuan Yu, Robert Doerning, Fei Xia, and Meliha Yetisgen. 2025. Morqa: Benchmarking evaluation metrics for medical open-ended question answering. *Preprint*, arXiv:2509.12405.

Cunxiang Wang, Sirui Cheng, Qipeng Guo, Yuanhao Yue, Bowen Ding, Zhikun Xu, Yidong Wang, Xiangkun Hu, Zheng Zhang, and Yue Zhang. 2023. Evaluating open-QA evaluation. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Xinyu Wang, Jijun Chi, Zhenghan Tai, Tung Sum Thomas Kwok, Muzhi Li, Zhuhong Li, Hailin He, Yuchen Hua, Peng Lu, Suyuchen Wang, and 1 others. 2025. Finsage: A multi-aspect rag system for financial filings question answering. *arXiv preprint arXiv:2504.14493*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Wutong Zhang, Hefeng Zhou, Qiang Zhou, Yunshen Li, Yuxin Liu, Jiong Lou, Chentao Wu, and Jie Li. 2025a. Towards comprehensive legal document analysis: A multi-round rag approach. In *Proceedings of the 2025 International Conference on Multimedia Retrieval*, pages 1840–1848.

Ze Yu Zhang, Zitao Li, Yaliang Li, Bolin Ding, and Bryan Kian Hsiang Low. 2025b. Respecting temporal-causal consistency: Entity-event knowledge graphs for retrieval-augmented generation. *arXiv preprint arXiv:2506.05939*.

A Prompt Templates

A.1 Metadata Extraction (SpecDB)

We use the following prompt for metadata extraction during SpecDB construction:

```
You are an expert at structured data
extraction. You will be given unstructured text
in markdown and must convert it into the given
structure.
Some rules to keep in mind:
1. version can be extracted by looking for
something of the form 'x.y.z' where x,y, and z
are numbers. It is prefaced by V or the word
Version:
2. docID can be extracted by looking for
something of the form 'az x.y' where x and y
are numbers and az represents any two letters.
This usually follows the phrase 3GPP. The docid
is the x.y
```

```
3. the timestamp is a date of the form YYYY-MM
4. Release is a number usually prefaced by the
word 'Release'
5. If you cannot instantiate a field, populate
it with the empty string.
```

A.2 CR-Specific Chunking (TDocDB)

We use the following prompt for CR-specific chunking during TDocDB construction:

```
You are an expert at extracting structured
change information from change request (CR)
markdown text.
Given the CR text, focus only on three
sections: 'summary of changes', 'reason for
change', and 'consequences if not approved'.
For each change mentioned in the 'summary of
changes' section produce one object with keys:
    - summary: the change text (single bullet or
    sentence)
    - reason: the portion(s) of 'reason for
    change' that explain why this change was
    introduced
    - consequence: the portion(s) of
    'consequences if not approved' that describe
    the impact if this change is not implemented
Output ONLY a JSON array (no markdown, no extra
commentary). If there are multiple bullets in a
section, map them in order; if mapping is
ambiguous, include best-effort contextual text.
```

A.3 Keyword Extraction (Specification-Evolution Reasoning)

We use the following prompt for direct keyword extraction during the specification-evolution reasoning phase:

```
You are an expert at extracting document
identifiers from unstructured text.
A document identifier (docID) typically follows
the format 'az x.y' where 'az' represents any
two letters and 'x.y' are numbers. It is
possible for the az to be omitted
This identifier is often found in proximity to
the phrase '3GPP'.
Your task is to identify and extract all the
docIDs from the provided text.
If no valid docID is present, return an empty
string.
```

A.4 Answer Generation

We use the following prompt for final answer generation:

```
Answer the following question in about 200
words.
- Explain the reasoning behind your answer.
- If possible, provide context, background
information, or insights that may enhance
understanding.
- Use the context below if it is relevant;
otherwise rely on general knowledge.

<context>
{context}
</context>

Question: {input}

Answer:
```

A.5 Pairwise Win Rate

We use the following prompt for the pairwise evaluator:

```
# Task
You are an expert in the 5G domain. You will be
provided with a 5G-related question, an
expert-written reference answer, and two
candidate answers to evaluate. Your task is to
determine which candidate answer is better
based on the reference answer and the following
criteria:
- **Answer Accuracy**: How factually correct is
each candidate answer compared to the reference
answer? Are there any inaccuracies, omissions,
or misleading statements?
- **Explanation Helpfulness**: How well does
each candidate answer explain the reasoning
behind itself? Does it provide useful context,
background information, or insights that
enhance understanding of the answer?

# Question
{question}

# Reference Answer
{ground_truth}

# Candidate Answers
- Answer A: {answer_a}
- Answer B: {answer_b}

# Instructions
Provide brief reasoning based on the criteria
above, followed by your final decision in the
format below:
```
Reasoning: <Your brief reasoning>
Judgment: <"Answer A" or "Answer B">
```
```

A.6 Rubric-based scoring

We use the following prompt to generate the initial question-specific for each QA pairs:

```
You are an expert in 5G telecommunications and
assessment design. Create a detailed scoring
rubric for the following question based on the
ground truth answer provided.

Question: {question}

Ground Truth Answer: {ground_truth}

Create a scoring rubric with a total of 5
points. Break down the points based on:
1. Core answer/concept (typically 2-3 points)
2. Key details/explanation (typically 1-2
points)
```

```
3. Technical accuracy (typically 0.5-1 point)
4. Completeness (typically 0.5-1 point)

Return ONLY a JSON object with this structure:
{
  "total_points": 5,
  "criteria": [
    {"description": "criterion description",
    "points": X},
    {"description": "criterion description",
    "points": Y}
  ]
}

Make the rubric specific to this question. Do
not include any other text outside the JSON.
```

After human curation and validation, we use the following prompt to score each answer with the question-specific rubrics:

```
You are an expert grader for 5G
telecommunications questions. Grade the
predicted answer against the ground truth using
the provided rubric.

Question: {question}

Ground Truth Answer: {ground_truth}

Predicted Answer: {predicted_answer}

Scoring Rubric (Total: {total_points} points):
{criteria_text}

Carefully evaluate the predicted answer against
each criterion in the rubric. Award points
based on:
- Factual correctness compared to ground truth
- Completeness of the answer
- Technical accuracy of terminology
- Whether key points from ground truth are
covered

Return ONLY a JSON object with this exact
structure:
{
  "score": X.X,
  "grading_reasoning": "Detailed explanation of
  the score, breaking down points
  awarded/deducted for each criterion"
}

The score should be a number between 0 and
{total_points}, and can include decimals (e.g.,
3.5, 4.0).
Be strict but fair. If the predicted answer
contradicts the ground truth or misses key
points, deduct points accordingly.
Do not include any text outside the JSON object.
```

A.7 CR QA Generation Prompt

We use the following prompt to generate question-answer pairs from approved Change Request documents. The prompt instructs the language model to create questions in one of two formats depending on the clarity of before-and-after states in the specification text, and to skip CRs that lack sufficient technical detail for meaningful QA generation.

```
You are analyzing a 3GPP Change Request (CR)
document. Based on the information provided,
generate ONE question-answer pair.

CR Information:
- Specification: {spec_number}
- Summary of Change: {summary}
- Reason for Change: {reason}
- Consequences if Not Approved: {consequence}
- Clauses Affected: {clauses_affected}

Decision rule BEFORE producing output:
- If AND ONLY IF the Summary of Change, Reason
for Change, AND Consequences are all very short
one-liners (i.e., each is so brief that you
cannot identify a meaningful technical change
and rationale), then DO NOT create a Q&A.
Instead, return:
  {
    "skip": true,
    "reason": "why the three fields are too
    simple to form a meaningful Q&A"
  }

Otherwise, create exactly ONE Q&A with the
following instructions.

Instructions for Q&A:
1) Decide if you can identify clear "before"
and "after" states in the spec text from the
provided info.
2) Generate a question in ONE of these formats:

   Format A (if before AND after states are
   clear; do NOT mention the spec number in the
   question):
   - Example templates:
     * "Why was [specific parameter/behavior X]
     changed to [specific parameter/behavior Y]?"
     * "What is the reason for changing
     [feature/requirement X] to
     [feature/requirement Y]?"
     * "I observed that [X] was modified to [Y].
     What is the rationale for this change?"

   Format B (if before OR after is unclear;
   explicitly mention the specification number):
   - Example templates:
     * "What is the reason for the change to
     [specific aspect] in specification
     {spec_number}?"
     * "Why was [feature/behavior] modified in
     3GPP TS {spec_number}?"
     * "I observed a change regarding [topic] in
     specification {spec_number}. What is the
     reason?"

3) The answer should:
   - Explain the reason for the change,
   - Include the consequences if the change is
   not accepted,
   - Be natural and technical (no copy-paste;
   use proper terminology).

Output format (strict JSON):
- If skipping:
```

```
  {
    "skip": true,
    "reason": "short explanation"
  }

- If generating Q&A:
  {
    "question": "your generated question",
    "answer": "your generated answer",
    "format_used": "A" or "B"
  }
```

All generated question-answer pairs undergo manual review to verify technical accuracy, ensure natural phrasing, and confirm that answers adequately explain both the rationale for the specification change and the potential consequences if the change were not approved.

A.8 Reference Helpfulness Filter

We use the following prompt to filter the helpful references for the microbenchmark on cross-reference resolution:

```
Compare the original chunk to the reference
chunk.
original chunk: {org_chunk}
reference chunk: {ref_chunk}
Instructions:
1. See if the reference chunk provides
information that would be useful to understand
what the original chunk is talking about.
2. Return 'helpful' if the reference chunk's
content will provide necessary detail to fully
understand what the original chunk's content is
talking about.
3. Return 'unhelpful' if the reference chunk's
content is not necessary to fully understand
what the original chunk is talking about.
4. Respond only with 'helpful' or 'unhelpful'
```

B Reference Extraction Patterns

We use the following regular-expression matcher in Python to extract references according to predefined patterns:

```
For reference extraction:
(clause\s+(\d+(.\d+)*).?(of \[\d+,
[A-Za-z0-9_\. ]*\])?)

(Table\s+(\d+([.\d+|\-\d])*).?)

(subclause\s+(\d+(.\d+)*))

(subclauses\s+(\d+(.\d+)*) and (\d+(.\d+)*))

For source document extraction:
\[\d+, [A-Za-z0-9_\. ]*\]

For specification identifier extraction
\b(?:TS\s*)?([0-9]{2}\.[0-9]{3})\b
```

C Volunteer Recruitment and QA Annotation Protocol

C.1 Recruitment Process

We recruit volunteers from doctoral and master's students in Computer Science programs. To ensure the technical depth needed for high-quality QA construction, candidates are required to have either (1) completed graduate-level coursework in mobile networks, wireless communications, or related telecommunications subjects, or (2) demonstrated practical experience with cellular network technologies through research projects, internships, or professional roles. Selected volunteers attended an orientation session covering project objectives, task scope, and quality standards. We emphasize rigorous technical accuracy and consistent application of annotation criteria across all materials. We do not anticipate that annotator demographics introduce bias into the dataset.

C.2 Annotator Instructions

C.2.1 Forum Post Curation (telecomHall)

Annotators review discussion threads to identify and curate clear question–answer exchanges related to 5G technology. For each candidate QA, they assess whether the question is well-posed and directly relevant to 5G topics (e.g., network architecture, protocols, physical layer specifications, core network functions, deployment). Questions that are ambiguous, overly broad, or only tangentially related to 5G are excluded. During the curation process, annotators verified that all retained QA pairs contain only technical content without personally identifying information, as the source materials consist of public technical discussions and educational documentation focused on telecommunications specifications rather than individual user information.

Corresponding answers are evaluated for technical correctness, drawing on annotators' domain expertise and, when necessary, consulting authoritative references (e.g., 3GPP specifications). Answers containing factual errors, outdated information, or misleading explanations are removed. Annotators also ensured completeness and clarity: each answer has to directly address the question, provide sufficient technical detail, and use precise language appropriate to the technical context. QA pairs are retained only if they satisfy all criteria.

C.2.2 Educational Content Annotation (ShareTechnote)

Annotators systematically read educational blog posts and convert key technical explanations into question–answer pairs. For each post, they identify central concepts and formulate clear, specific questions reflecting realistic information needs of students or professionals studying 5G (e.g., technical specifications, operational principles, system components, practical applications).

Answers are composed directly from the source material, designed to be self-contained and technically accurate. Annotators verified alignment with current 5G standards and cross-checked unclear points against authoritative sources before finalizing the pairs. As with forum curation, only QA items meeting standards for accuracy, completeness, and clarity are included in the dataset.

The curation, data collection, and verification process require approximately 120 hours of annotation effort across a 3-month collection period.

D Comparisons with Telco-oRAG

To evaluate whether external web retrieval could compensate for a lack of standard-native reasoning, we also benchmarked Telco-oRAG on a representative subset of our LLMs (GPT-4o, Qwen3-8B, and Qwen3-32B). We run Telco-oRAG on real-world and CR-focused QA using the same corpus (3GPP Rel-18), retrieval budget (k=10 chunks) and prompt.

D.1 Results on Real-World QA

Table 5 presents a comparison of Telco-oRAG against Chat3GPP and DEEPSPECS. DEEPSPECS shows significant gains against Telco-oRAG across all models. Against Qwen3-32B, Telco-oRAG shows performance degradation against base model.

D.2 Results on CR-Focused QA

Table 6 presents a comparison of Telco-oRAG against Chat3GPP and DEEPSPECS. DEEPSPECS shows significant gains against Telco-oRAG across all models. Against Qwen3-32B, Telco-oRAG shows performance degradation against base model.

Table 5: Comparative performance on the real-world QA dataset. All pairwise comparisons are made against a GPT-4o baseline. Note that the win rate for GPT-4o is calculated from a self-to-self comparison and therefore always 50%.

Model	Method	Length (tokens)	Win Rate vs. GPT-4o (%)
GPT-4o	Base Model	295	(50)
	Telco-oRAG	287	51.0
	Chat3GPP	295	62.5
	DEEPSPECS	301	**68.1**
Qwen3-8B	Base Model	346	50.8
	Telco-oRAG	404	51.7
	Chat3GPP	490	74.4
	DEEPSPECS	483	**79.8**
Qwen3-32B	Base Model	373	73.6
	Telco-oRAG	430	69.9
	Chat3GPP	499	82.9
	DEEPSPECS	497	**88.0**

Table 6: Comparative performance on the CR-Focused QA Dataset. Note that DEEPSPECS has reference resolution disabled.

Model	Method	Length (tokens)	Win Rate vs. GPT-4o (%)
GPT-4o	Base Model	273	(50)
	Telco-oRAG	262	53.9
	Chat3GPP	265	59.3
	DEEPSPECS	269	**77.0**
Qwen3-8B	Base Model	312	73.3
	Telco-oRAG	382	74.7
	Chat3GPP	466	85.7
	DEEPSPECS	427	**92.6**
Qwen3-32B	Base Model	353	89.9
	Telco-oRAG	440	86.0
	Chat3GPP	549	93.7
	DEEPSPECS	506	**95.7**

E Additional Results on Real-World QA

E.1 Dataset Categorization

The 573 questions in our real-world QA dataset are organized into five technical categories that reflect the organizational structure of 5G standardization and engineering practice:

- **Category 1: Air Interface & Physical Layer** (186 questions, 32.5%) – Physical channel structures, modulation schemes, MIMO configurations, beamforming, and radio resource allocation in the NR air interface.
- **Category 2: Protocol Stack & Resource Management** (174 questions, 30.4%) – Layer 2/3 protocol operations (MAC, RLC, PDCP,

Table 7: Pairwise win rate (%) on the real-world QA dataset for Qwen backends under non-thinking and thinking modes (vs. GPT-4o). Thinking mode generally improves performance across methods, and DEEPSPECS consistently outperforms Chat3GPP in both modes.

Model	Method	Non-thinking (%)	Thinking (%)
Qwen3-4B	Base Model	20.6	34.4
	Chat3GPP	57.0	67.3
	DEEPSPECS	62.3	71.2
Qwen3-8B	Base Model	34.2	50.8
	Chat3GPP	63.9	74.4
	DEEPSPECS	67.1	79.8
Qwen3-14B	Base Model	46.9	64.7
	Chat3GPP	69.1	78.9
	DEEPSPECS	72.4	83.1
Qwen3-32B	Base Model	55.8	73.6
	Chat3GPP	73.0	82.9
	DEEPSPECS	77.2	88.0

SDAP), quality-of-service mechanisms, numerology, and resource mapping procedures.

- **Category 3: Core Network & Session Management** (114 questions, 19.9%) – 5G Core architecture, session establishment and mobility procedures, network function interactions, and signaling flows.
- **Category 4: Network Architecture & Deployment** (66 questions, 11.5%) – Network slicing, multi-access architectures, deployment scenarios (NSA/SA), interworking with legacy systems, and infrastructure planning.
- **Category 5: Operations, Optimization & Troubleshooting** (33 questions, 5.8%) – Performance optimization strategies, diagnostic procedures, measurement reporting, and operational best practices.

The distribution reflects the relative emphasis in 5G technical documentation and practitioner discussions, with lower-layer protocol mechanics and physical-layer operations receiving the most attention. Each question is assigned to exactly one category by the annotation team based on its primary technical focus.

E.2 Thinking vs. Non-thinking Mode

Table 7 compares Qwen backends under non-thinking and thinking modes on the real-world QA dataset. Thinking mode generally improves win rates across methods, and DEEPSPECS consistently outperforms Chat3GPP in both modes.

Table 8: Rubric-based scoring results comparing DEEPSPECS against base models. Scores range from 0 to 5. All differences significant at p<0.0001 (Wilcoxon signed-rank test).

Metric	GPT-4.1	GPT-4.1-mini
Baseline Mean	2.67	2.46
DEEPSPECS Mean	**3.29**	**3.06**
Improvement	+0.62***	+0.60***
Effect Size (dz)	0.41	0.40

E.3 Additional Evaluation

Table 8 presents additional rubrics-scoring results. Table 9 presents the Per-category win rates against the GPT-4o baseline across five 5G technical domains, demonstrating the performance of base models, Chat3GPP, and DEEPSPECS for different model families and sizes.

F Additional Results on CR-focused QA

Table 10 presents additional pairwise win rate results. Table 11 presents additional rubric-based scoring results.

G Microbenchmark: Cross-Reference Resolution

We quantify the gap between semantic retrieval and cross-reference resolution using a microbenchmark over 10 specifications (TS 38.181–TS 38.304 from release 18). In total, we sample 547 chunks and extract 903 cross-references. These are filtered with an LLM helpfulness judge (Appendix A.8) and validated by experts to form a set of helpful references, on which we calculate the precision, recall, and f1 score. We compare the retrievals of (a) Chat3GPP and (b) DEEPSPECS's standalone cross-reference resolver to this set of helpful references. To ensure a fair comparison, we fix retrieval parameters (k_1=3, k_2=2, k_3=0) for DEEPSPECS and have Chat3GPP return the top-5 chunks. Table 12 shows substantially higher precision, recall and F1 for DEEPSPECS, indicating the limitations of semantic-only retrieval on interlinked specs and the effectiveness of DEEPSPECS's cross-reference resolution method.

H Implementation Details

We provide comprehensive details of our implementation setup to facilitate reproducibility. This includes software dependencies, model configu-

Table 9: Per-category performance on the real-world QA dataset (Win Rate vs. GPT-4o, %).

Model	Method	Air Interface	Protocol Stack	Core Network	Network Arch.	Operations & Troubleshooting	Overall
GPT-4o	Base Model	(50)	(50)	(50)	(50)	(50)	(50)
	Chat3GPP	54.8	67.8	64.5	62.1	69.7	62.4
	DEEPSPECS	**60.5**	**72.1**	**72.4**	**66.7**	**77.3**	**68.1**
Qwen3-4B	Base Model	33.6	30.7	38.6	39.4	33.3	34.4
	Chat3GPP	59.9	70.4	71.9	72.0	66.7	67.3
	DEEPSPECS	**62.9**	**76.1**	**74.6**	**75.0**	**72.7**	**71.2**
Qwen3-8B	Base Model	47.6	50.6	50.4	56.1	60.6	50.8
	Chat3GPP	72.0	78.4	71.9	74.2	75.8	74.4
	DEEPSPECS	**73.4**	**83.6**	**80.3**	**86.4**	**80.3**	**79.8**
Qwen3-14B	Base Model	56.5	61.5	65.4	65.2	63.6	61.2
	Chat3GPP	73.9	**85.9**	79.8	75.8	77.3	79.1
	DEEPSPECS	**75.8**	85.6	**82.0**	**87.9**	**81.8**	**81.8**
Qwen3-32B	Base Model	67.7	77.0	71.9	78.0	86.4	73.6
	Chat3GPP	79.6	87.4	84.2	78.0	83.3	82.9
	DEEPSPECS	**82.3**	**92.5**	**86.8**	**92.4**	**92.4**	**88.0**
Claude 3.5 Haiku	Base Model	47.8	50.9	53.1	41.7	39.4	48.6
	Chat3GPP	51.6	62.4	57.5	53.0	59.1	56.6
	DEEPSPECS	**53.0**	**64.4**	**61.8**	**58.3**	**63.6**	**59.4**
GPT-4.1 mini	Base Model	86.3	81.3	81.1	75.0	78.8	82.0
	Chat3GPP	84.4	89.1	**89.9**	88.6	90.9	87.8
	DEEPSPECS	**87.6**	**91.4**	89.5	**92.4**	**93.9**	**90.1**
GPT-4.1	Base Model	89.2	87.6	83.8	84.8	89.4	87.2
	Chat3GPP	92.5	93.4	94.3	**97.7**	92.4	93.7
	DEEPSPECS	**93.8**	**95.4**	**96.1**	**97.7**	**97.0**	**95.4**

Table 10: Additional results on the CR-focused QA dataset.

Model	Method	Length (tokens)	Win Rate vs. GPT-4o (%)
Qwen3-4B	Base Model	283	68.6
	Chat3GPP	372	81.4
	DEEPSPECS	341	**88.0**
Qwen3-8B	Base Model	312	73.3
	Chat3GPP	466	85.7
	DEEPSPECS	427	**92.6**
Qwen3-14B	Base Model	295	81.1
	Chat3GPP	481	91.4
	DEEPSPECS	443	**93.6**
Qwen3-32B	Base Model	353	89.9
	Chat3GPP	549	93.7
	DEEPSPECS	506	**95.7**

Table 11: Rubric-based scoring results on CR-focused QA using GPT-4o as the base model. Scores range from 0 to 5. All pairwise differences significant at p<0.0001 (Wilcoxon signed-rank test).

Metric	Score
Base Model Mean	2.26
Chat3GPP Mean	2.48
DEEPSPECS Mean	**3.05**
DEEPSPECS vs. Base	+0.79 (0.68) ***
DEEPSPECS vs. Chat3GPP	+0.58 (0.51) ***
Chat3GPP vs. Base	+0.21 (0.26) ***

Effect sizes (dz) shown in parentheses.

rations, inference hyperparameters, and computational resources.

H.1 Software Environment

All experiments were conducted in Python 3.11.11. Table 13 lists the key software packages and versions used in our implementation.

H.2 Inference Hyperparameters

We adopt default or recommended settings for inference across APIs and locally deployed models:

- **OpenAI models (OpenAI, 2024, 2025):** temperature = 1.0, top_p = 1.0
- **Claude 3.5 Haiku (Anthropic, 2024):** temperature = 1.0, top_p = 1.0
- **Qwen3 models (Yang et al., 2025):** temperature = 0.6, top_p = 0.95, top_k = 20, with

Table 12: Cross-reference resolution microbenchmark performance. Precision, recall, and f1 are calculated on the filtered set of "helpful" references.

Method	Precision	Recall	F1
Chat3GPP	0.0709	0.2334	0.1031
DEEPSPECS	**0.1876**	**0.7055**	**0.2837**

Package	Version
Core Environment	
Python	3.11.11
pandas	2.2.3
PyYAML	6.0.2
pydantic	2.10.6
tqdm	4.67.1
gradio	5.13.1
Retrieval	
openai (Python client)	1.97.1
ChromaDB	0.6.3
langchain	0.3.15
langchain-core	0.3.31
langchain-openai	0.3.2
langchain-community	0.3.15
langchain-text-splitters	0.3.5
tiktoken	0.8.0
Parsing	
BeautifulSoup	0.0.2
python-docx	1.2.0
MarkItDown	0.1.2
Model Inference	
transformers	4.56.2
PyTorch	2.8.0
vLLM	0.8.5

Table 13: Software packages and versions.

thinking mode enabled

H.3 Computational Infrastructure

Hardware and API Access. Experiments with GPT and Claude models were conducted via official APIs. For Qwen3 models, inference was performed locally using `vLLM` on a single NVIDIA A100 80GB GPU.

Budget and Compute Usage. The estimated total expenditure on LLM API calls was approximately $500. Local inference with Qwen3 models required around 30 GPU hours in total.

H.4 Other Experimental Settings

Unless otherwise specified:

- DEEPSPECS cross-reference resolution, the recursion depth is fixed at 2.
- All embeddings are obtained using `text-embedding-3-large`.
- GPT-4o serves as the default assistant, extractor, generator, and evaluator model.

I An Illustrative Example: Specification-Evolution Reasoning

DEEPSPECS uses clause-level cross-reference resolution and specification-evolution reasoning to embed structural and temporal insights on the 5G Domain in its retrieval process. This allows it to better answer questions that require such insights. Below, we explore how specification-evolution reasoning happens with a clear example.

> **Q**: What was the specific reason for clarifying the parameter range of the quantity P_f?

This is the type of question that a user looking at the change history of specification 38.211 may have. The technical specifications are purely prescriptive and would not contain this motivating information. Pure semantic search in TDocDB would encounter a lot of noise. DEEPSPECS attempts to address this issue using the results of a semantic search in ChangeDB to build filters to constrain the subsequent TDocDB search.

Below is one of the returned chunks from ChangeDB, with a partially omitted metadata section.

> The quantity P_f belongs to $\{2, 4\}$ is given by the higher-layer parameter FreqScalingFactor if configured, otherwise
> **Metadata:** docID - 38.211, from_version - 17.1.0, to_version - 17.2.0

This chunk represents the addition of this text to version 17.1.0 of specification 38.211. So, the semantic search returns the referenced change. We use this to build a filter that looks for only the chunks about the specification with docID 38.211 in the following semantic search through the TDocDB.

Below is one of the returned chunks from the semantic search in TDocDB. We have partially omitted some of the metadata for clarity.

> **Change Summary:** Added the correct parameter range.
>
> **Reason:** The range of for SRS is missing (R1-2205261).
>
> **Consequence if this change is not accepted:** Unclear parameter range.
>
> **Metadata:** docID - 38.211, version - 17.1.0, ...

Our unique chunking strategy for TDocs extracts information on what the change is (which aids semantic searches), why the change was made, and

the consequences if the change was not improved. All of this is highly relevant to the user's original query.

J An Illustrative Example: End-to-End Workflow Demonstration

To demonstrate our end-to-end QA workflow in practice, we attach a detailed case analysis of DEEPSPECS processing a complex real-world query.

Question: *"If a UE cannot comply with an RRCReconfiguration message, which procedure should it initiate, and why was the rollback reference tied to the reception of the message?"*

Stage 1: Semantic Retrieval. Standard semantic retrieval from SpecDB returns (among others):

> "... if the UE is unable to comply with an RRCReconfiguration ... it shall continue using the configuration used prior to the reception of RRCReconfiguration message; ... otherwise initiate the connection re-establishment procedure as specified in TS 38.331 [10], clause 5.3.7; ..."

Stage 2.1: Cross-Reference Resolution. This chunk cites clause 5.3.7 for the connection re-establishment procedure. We extract the reference (TS 38.331, clause 5.3.7) via a regex-based extractor, build a clause-level metadata filter, and run a secondary SpecDB retrieval (same as Stage 1) to obtain the exact procedure definition.

Stage 2.2: Spec-Evolution Reasoning. In parallel, since the question does not mention a specific document ID, we semantically search ChangeDB and retrieve a top change:

> [TS 38.331, v16.3.1→16.3.2, Clause 5.3.5.8.2, addition] "... configuration used prior to the reception of RRCReconfiguration message..."

From this, we extract the document ID 38.331 and build a metadata filter to narrow the retrieval space in TDocDB, yielding the following rationale:

> **Change Summary:** ... the UE continues using the configuration it had before the failed compliance check.
> **Reason:** ... the compliance check may be performed at execution... the UE shall continue using the configuration it had before the reception of the message.
> **Consequence if this change is not accepted:** ... UE/network misalignment...

Crucially, this rationale is prescriptive and exists only in the Change Request documentation, not in the specification text itself.

Stage 3: Answer Synthesis. We pass the combined contexts from Stages 1, 2.1, and 2.2 to the LLM generator. The key historical context and procedural definitions needed to answer the "which procedure" and "why tied to" portions of the query only appear in the payloads from Stages 2.1 and 2.2, allowing DEEPSPECS to produce a comprehensive, expert-level answer.