

TelecomRAG: Taming Telecom Standards with Retrieval Augmented Generation and LLMs

Girma M. Yilma

NEC Laboratories Europe, Germany
girma.yilma@neclab.eu

Andres Garcia-Saavedra

NEC Laboratories Europe, Germany
andres.garcia.saavedra@neclab.eu

Jose A. Ayala-Romero

NEC Laboratories Europe, Germany
jose.ayala@neclab.eu

Xavier Costa-Perez

i2CAT, NEC Laboratories Europe, and ICREA, Spain
xavier.costa@neclab.eu

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.

The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

Large Language Models (LLMs) have immense potential to transform the telecommunications industry. They could help professionals understand complex standards, generate code, and accelerate development. However, traditional LLMs struggle with the precision and source verification essential for telecom work. To address this, specialized LLM-based solutions tailored to telecommunication standards are needed. This *Editorial Note* showcases how Retrieval-Augmented Generation (RAG) can offer a way to create precise, factual answers. In particular, we show how to build a Telecommunication Standards Assistant that provides accurate, detailed, and verifiable responses. We show a usage example of this framework using 3GPP Release 16 and Release 18 specification documents. We believe that the application of RAG can bring significant value to the telecommunications field.

CCS CONCEPTS

- Information systems → Language models;
- Networks → Network design principles;

KEYWORDS

Telecommunications, LLM, Standards, 3GPP, O-RAN, ETSI

1 INTRODUCTION

Large Language Models (LLMs), built upon the popular Transformer architecture and massive training datasets, are revolutionizing the way we interact with language. Their ability to “understand” complex text, generate responses, and translate between languages has far-reaching implications across industries. Among these, the telecommunications domain stands to benefit significantly from the advancements in LLMs.

Telecommunications professionals are tasked with navigating the intricate and evolving landscape of standard specification documents established by bodies such as 3GPP, ETSI, and O-RAN. These standards define the fundamental technologies that enable global connectivity and interoperability. AI-powered assistants based on LLMs have the potential to revolutionize how professionals interact with these standards. Such assistants could rapidly analyze dense specifications, generate code snippets, assist in debugging, and

even offer insights that bridge the gap between different standards. This holds the promise of increased efficiency, innovation, and the streamlined development of robust telecom solutions.

However, conventional LLM-based assistants often struggle with knowledge-intensive tasks, a point discussed in detail in Section 2 of this paper. While LLMs can store factual information implicitly within their vast parameter sets, this knowledge is not easily accessible, updated, or traceable. In the context of telecommunications, where precision and reliance on standard-specified sources are critical, this limitation creates hurdles for widespread adoption in the industry.

To overcome these challenges, there is a growing need for specialized LLM-based solutions tailored to telecommunication standards. Section 2 explores various domains where LLMs have shown success, highlighting the need for such adaptation within telecommunications. Retrieval-Augmented Generation (RAG), a technique that combines LLM capabilities with a non-parametric knowledge base, offers a promising avenue for addressing the need for precise and factual answers. Accordingly, we discuss in Section 3 the challenges of applying this approach specifically to telecommunication standards, followed by the design of an architecture that tackles these challenges to build a Telecommunication Standards Assistant that is accurate, technically deep, and provides verifiable answers.

In Section 4, we present an implementation of this framework. It leverages APIs and libraries to ingest telecom standards documents, create knowledge embeddings, and seamlessly integrate them within a user-friendly interface. Finally, we show an operational example of the framework. We corroborate that the output of the system is technically precise and verifiable when comparing with ground truth extracted directly from the standard documents.

2 LLMS AND LANGUAGE GENERATION

This section provides some background on LLMs, their abilities (Sec. 2.1), and LLMs for specific application domains (Sec. 2.2). Then, we review the related work retrieval-augmented methods (Sec. 2.3), which are the base of the proposed framework.

2.1 Background on LLMs

Large language models (LLMs) are Transformer-based language models with hundreds of billions or more parameters trained on an enormous corpus of text. Some examples include Gemini, LLaMa, and GPT-4. These models demonstrate significant capabilities in comprehending natural language and tackling intricate tasks through text generation. Although existing LLMs adopt the same Transformer architecture and pre-training objectives as other smaller language models, the significant increase in the model size, data size, and computing capacity largely improves the performance of LLMs. In fact, some works propose different scaling laws of LLMs, showing that model performance has strong dependencies with these three dimensions [6].

One of the most prominent features of LLMs is the emergent abilities, formally defined as the abilities that are not present in smaller models but arise in larger ones. The authors in [14] show several examples of emergent abilities, where the performance in a task via few-shot prompting is random until a particular scale. When the model size surpasses a certain scale the performance increases well above random. This ability, also referred to as *in-context learning*, is formally introduced by GPT-3 [2] but is not shown by previous models such as GPT-1 and GPT-2. Another ability that emerged in large models is the step-by-step reasoning also called *chain of thoughts* [13]. This strategy is very effective in solving complex tasks that involve multiple steps such as math problems.

As emergent abilities cannot be predicted through the scaling laws of LLMs, this raises the question of whether additional scaling could potentially further expand the range of capabilities of language models [14].

2.2 LLMs for specific application domains

LLMs have been applied to different specific domain areas. In healthcare, Med-PaLM [10] achieves expert-level performance on the United States Medical Licensing Examination (USMLE). It relies on the pre-trained PaLM model enhanced with several techniques, namely, few-shot prompting, chain-of-thought and self-consistency prompting, and prompt tuning. However, the use of LLMs in healthcare comes with the risk of fabricating medical misinformation (e.g., suggesting advice inconsistent with medical guidelines) [5].

Finance is another important field where the application of LLMs is promising. In this field, BloombergGPT [15] is a proprietary model based on BLOOM architecture trained with a mix of general purpose and specialized finance data. This results in a model that achieves SoTA results in financial benchmarks, while keeping competitive performance on general-purpose tasks. As an alternative, FinGPT is an open-source framework comprising several layers for data gathering and engineering, pre-trained LLM models (e.g., LLaMA), and different fine-tuning techniques such as Fine-tuning via Low-rank Adaptation (LoRA) and Reinforcement Learning from Human Feedback (RLHF).

In the domain of scientific research, LLMs specially pre-trained on scientific-related corpora have been proposed (e.g. Galactica [11], Minerva [9]). These models aim to assist in different stages of the scientific research pipeline.

However, all these solutions have the potential risk of hallucination, i.e., the generation of content that is not grounded in factual information but is instead a product of the model's creative extrapolation. This phenomenon occurs even when the models are trained with a high-quality and highly curated corpus of data, as in the case of Galactica [11]. Moreover, hallucination may have serious implications in critical domains such as healthcare [5]. In order to reduce the hallucination phenomenon and provide reliable factual information, other LLM-based approaches have been proposed as explained in the next section.

2.3 Retrieval-Augmented Methods

LLMs have demonstrated the ability to retain factual information within their parameters and excel in achieving state-of-the-art performance when fine-tuned for natural language processing tasks. However, the acquired world knowledge is implicitly encoded within the parameters of the underlying neural network, which poses a challenge in discerning what knowledge is stored in the network and where. Consequently, LLMs have serious limitations to *i*) access and precisely manipulate knowledge; *ii*) add new information or include updates to the current knowledge. Thus, task-specific architectures can outperform language models in knowledge-intensive tasks¹ [4, 8].

To improve the performance in knowledge-intensive tasks, some works in the literature propose hybrid models that combine a non-parametric memory (e.g., a textual knowledge corpus such as Wikipedia) with a pre-trained parametric memory (e.g., a LLM) [4, 8]. This architecture retrieves relevant information from the non-parametric memory, which is later processed and interpreted by the parametric memory (usually a language model), resulting in more factual and specific answers. Moreover, this architecture allows us to directly modify or expand the knowledge stored in the parametric memory without having to modify the parametric memory.

REALM [4] augments a masked language model with a differentiable retriever, which gathers knowledge from a textual knowledge corpus. Both the language model and the retriever are trained together end-to-end according to the language model objectives. This strategy poses a significant computational challenge since the backpropagation through the retriever must consider millions of documents in the textual knowledge corpus. Moreover, REALM only explores open-domain extractive question answering, i.e., its objective is to predict masked tokens in a sentence.

The authors in [8] expand this idea by proposing the retrieval-augmented generation (RAG) methodology for sequence-to-sequence (seq2seq) language generation. RAG considers a retriever based on Dense Passage Retrieval (DPR) [7] and a seq2seq model (e.g., BART architecture) for language generation. The DPR retriever follows a bi-encoder architecture, one for the queries and the other for the documents in the knowledge corpus. This method can be fine-tuned on any seq2seq task, whereby both the generator and retriever are jointly learned. In the fine-tuning phase, only the query encoder is trained. This reduces considerably the computational burden compared to REALM, which periodically updates the document index

¹This term refers to tasks that humans could not reasonably be expected to accomplish without relying on an external knowledge source.

during pre-training [4]. The results in [8] show that RAG responses are more factual, specific, and diverse than other baselines based on language models. For these reasons and given the nature of the task involved in telecommunication standards, we propose a framework based on RAG detailed in the next section.

3 TELECOMRAG DESIGN

This section details the conceptual design of the framework, referred to as TelecomRAG. It mainly relies on RAG with some practical modifications. This solution combines information retrieval techniques and language models. As detailed in the previous section, this combination is more effective than language models alone in knowledge-intensive tasks by providing more factual answers. Moreover, there are several advantages of generating answers instead of just extracting them directly from the knowledge base. For example, we may need to combine and elaborate the information included in several documents to obtain the answer to a specific query. Also, sometimes documents can contain clues about an answer but not the literal response.

TelecomRAG comprises two stages. In the first stage, we generate a vector database from the Telecom knowledge base. This stage is executed offline before launching the system. The vector database can be updated whenever new files need to be considered (e.g., new 3GPP releases or document updates). The second stage is executed for every user query and establishes the conversational question-answering pipeline.

3.1 Generation of Telecom vector database (Offline stage).

The vector database creation involves the generation of embedding representations of the files in the Telecom knowledge base. Embeddings are dense vector representations in a latent space and they aim to capture the semantic meaning of the content. Thus, the embedding of two different documents should be close in the vectorial space if the content of these documents is related (even if they use completely different words). Based on this, documents with content related to a query can be retrieved and the system's output can be grounded on factual information contained in these documents.

As an initial step, the files in the Telecom Knowledge Base need to be cleaned and chunked. The configuration of the chunking is crucial. Overly short chunks provide insufficient information, while excessively long passages hinder the creation of a rich semantic space and may exceed the LLM's context window limit. The documents are divided into passages (referred to as *documents* henceforth) of a fixed number of words with an overlap between consecutive chunks. For each document, one embedding is computed and stored in the vector database. A common practice to compute the embedding is to use a language model based on the BERT architecture [3, 8].

3.2 Document Retrieval and Presentation (Online stage).

The pipeline of the online stage is depicted in Fig. 1, comprising several building blocks: *i*) the retriever, which processes the input query, gathers related documents from the knowledge base, and

outputs the LLM input; *ii*) the history module, which stores past queries and responses; *iii*) the response generation and optimization module, which receives the LLM input and generates the final output. In the following, we detail the steps of the online pipeline.

First, the *retriever* generates the *standalone query* based on the user's new query and the history of past queries and responses. It is important for the standalone query to be self-contained for the retrieval step. If we only use the new query, relevant context content may be missing. Conversely, if we input the full past conversation, irrelevant information may hinder an accurate retrieval. Therefore, the standalone query is generated by an LLM taking as input the new user's query and the history provided by the *history module*.

Second, the embedding of the standalone query is computed using an encoder. We use the same encoder as in the offline stage allowing the documents and the queries to be encoded in the same latent semantic space.

Third, we perform a semantic search in the vector database. We look for document embeddings that are similar to the encoded query based on a similarity metric. As mentioned before, the embeddings capture the “gist” of the text regardless of its length or the specific wording. Given that the dimensionality of the embeddings is usually high and the vector database can contain a large number of embeddings, an efficient search algorithm is needed to reduce the computational burden. We retrieve K documents from the knowledge base whose embeddings obtained the highest similarity metric value.

Fourth, we compose the *LLM input* with the standalone query, the K retrieved documents, and the history of queries and outputs, as shown in Fig. 1. The LLM input is passed to the Base LLM in the *response generation & optimization* module.

Fifth, the Base LLM generates a *tentative output* based on the LLM input. To adapt the answer to different contexts, we use role-playing prompting techniques. In some cases, we may need a very detailed and comprehensive answer. To this end, we can assign the LLM the role of a “standard expert” and the user the role of a “non-experienced trainee”. Based on this, we can generate a prompt as follows: “*Assume you are a 3GPP standard expert and need to provide a very comprehensive answer to a non-experienced trainee.*” Thus, the tone and verbosity of the output can be easily adjusted.

Sixth, the tentative output generated by the Base LLM is checked by the *verification & optimization module*. The goal of this module is to check the appropriateness of the output. For example, if no documents are retrieved, a predefined message is sent to the user informing that there are no documents in the knowledge base related to the input query. Also, improper outputs can be filtered based on keywords. In other cases, the user can actively participate in the verification & optimization loop by, for example, improving the query or discarding irrelevant parts of the retrieved documents or history manually. Once the verification & optimization is passed, the system outputs the final response.

4 IMPLEMENTATION AND EXAMPLE OF OPERATION

In the following, we present our implementation of the above design (Section 4.1) and then, we showcase the operation of the framework using an example query (Section 4.2).

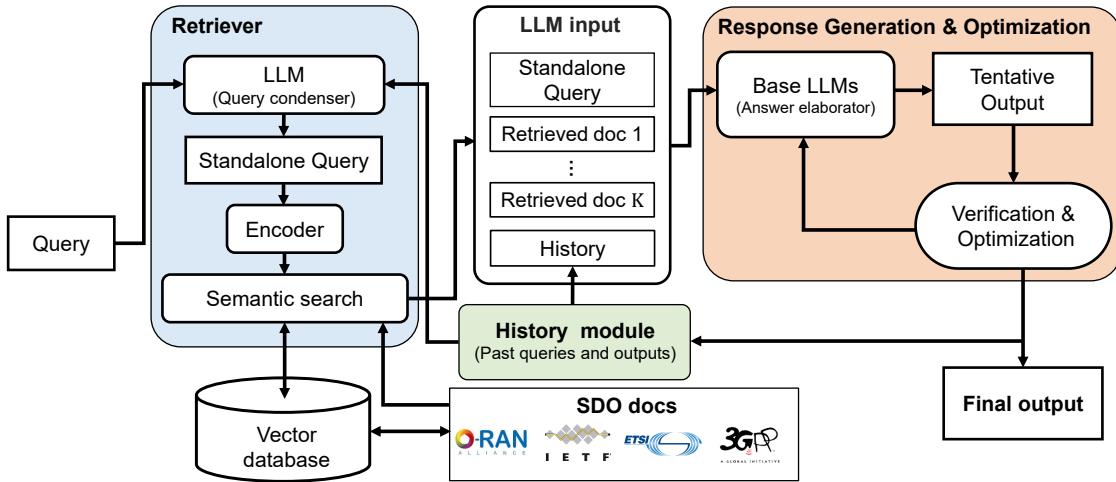


Figure 1: TelecomRAG: Architecture of the RAG-based Telecommunication Standards Assistant.

Table 1: ECN Failure Indication data from the correct 3GPP specification document [1, Section 8.30].

Procedure	Initiated	Information element name	Information element required	Information element description
ECN Failure Indication	IMS-AGW	Context	Mandatory	This information element indicates the context for the bearer termination.
		Bearer Termination	Mandatory	This information element indicates the bearer termination for which the ECN failure is reported.
		ECN Error Indication	Mandatory	This information element indicates an ECN failure event.
ECN Failure Indication Ack	IMS-ALG	Context	Mandatory	This information element indicates the context where the command was executed.
		Bearer Termination	Mandatory	This information element indicates the bearer termination where the command was executed.

Table 2: What are the information elements included in the “ECN Failure Indication”, and how are they defined?

TelecomRAG's response	<p>The ECN Failure Indication procedure, as described in the context of 3GPP standards, is used to report ECN (Explicit Congestion Notification) related failures. This procedure is part of the communication between IMS-ALG (IP Multimedia Subsystem - Application Layer Gateway) and IMS-AGW (IP Multimedia Subsystem - Access Gateway). The specific information elements involved in the ECN Failure Indication procedure are as follows:</p> <ol style="list-style-type: none"> (1) Context: This mandatory information element indicates the context for the bearer termination. The context provides the necessary background or situation associated with the bearer that is being terminated. (2) Bearer Termination: This mandatory information element indicates the bearer termination for which the ECN failure is being reported. The bearer termination refers to the point where a particular data session or communication is being terminated. (3) ECN Error Indication: This mandatory information element indicates an ECN failure event. An ECN failure event is a situation where the ECN mechanism has detected congestion in the network, but due to some error, the congestion notification could not be conveyed or processed as expected. <p>In the ECN Failure Indication procedure, when the IMS-AGW receives any IP headers indicating ECN Congestion Experienced (ECN-CE), it should trigger rate adaptation in accordance with the specifications given in 3GPP TS 26.114. If there is an ECN failure, the IMS-AGW reports this failure to the IMS-ALG using the ECN Failure Indication message containing the aforementioned information elements.</p> <p>Additionally, the ECN Failure Indication Acknowledgment (Ack) is used to acknowledge the receipt of the ECN Failure Indication. It also contains the Context and Bearer Termination information elements, which indicate the context and bearer termination where the command was executed.</p> <p>References: “IP Multimedia Subsystem (IMS) Application Level Gateway (IMS-ALG) - IMS Access Gateway (IMS-AGW) interface” 3GPP Technical Specification 23.334</p>			

4.1 Implementation

For the implementation of the RAG-based Telecommunication Standard assistant, we opted for Langchain [12], a framework that provides an array of built-in wrappers and utilities designed to extract reasoning capabilities from language models. Langchain provides most of the needed functionalities to implement the framework, it is well-documented and allows an agile implementation.

The first step is to build the knowledge base of standards. In our example, we use the complete set of 3GPP release-16 and

release-18 standard specification documents. These documents are in various formats (PDF, TXT, DOCX, DOC), which can be readily processed by the `DirectoryLoader` method from the library `Langchain.document_loaders`, along with custom functions for progress tracking, multi-threading, and error handling (e.g., corrupted file formats).

We divided the data into documents of 4000 characters with a 100-characters overlap between consecutive chunks using the `CharacterTextSplitter` method from `Langchain.text_splitter`.

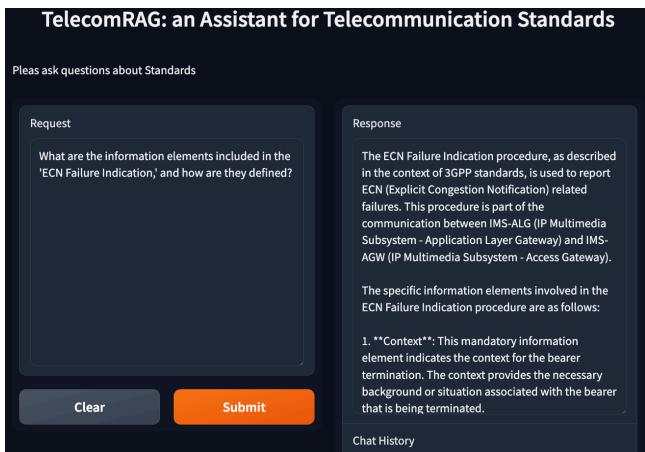


Figure 2: Graphical User Interface of TelecomRAG, an Assistant for Telecommunication Standards.

For every document, we then generated an embedding using the OpenAI’s `text-embedding-ada-002` embedding model through the `Langchain.embeddings.openai` library. This embedding model is also used to encode the standalone query at the retriever. After all the text documents are encoded, we store the embeddings in a *vector database* using `Chroma`, an AI-native vector database from `Langchain.vectorstores`.

We implement the pipeline of the online stage using the method `ConversationalRetrievalChain` from `Langchain.chains` library. We use the OpenAI API to query an LLM (GPT model) for the query condensation (at the retriever) and for the answer elaboration (at the response generation & optimization module). In our example, we use `gpt-4-1106-preview`.

We use *cosine similarity* as the similarity metric in our system. Cosine similarity is a computationally efficient solution that has proven good performance in this type of application. As vector datasets grow, semantic searches become increasingly computationally expensive. Consequently, approximate search algorithms are used to balance performance and speed in the search. In our case, we use an approximate nearest neighbor (ANN) algorithm called Hierarchical Navigable Small World (HNSW) implemented in `Chroma`. We consider the $K = 4$ documents with the highest cosine similarity obtained by HNSW algorithm to be included in the LLM input.

We wrapped up the system’s pipeline into a user-friendly web interface using `Gradio` that resembles that of conventional AI chat-bots such as `Gemini` or `ChatGPT`, as shown in Fig. 2.

4.2 Example of operation

We evaluated TelecomRAG on a large set of technical questions derived from 3GPP specifications. The assistant consistently provided thorough, accurate, and technically detailed responses useful for telecom engineers, including references to specific 3GPP documents for verification.

We showcase the operation of the framework with the query “*What are the information elements included in the ‘ECN Failure Indication’, and how are they defined?*”, which is described in [1]

(Section 8.30) and summarized in Table 1 for validation. Table 2 shows the response provided by TelecomRAG, including a reference to the right 3GPP document. We can observe that the response shows accuracy and verifiability.

Finally, we issue the same queries to general-purpose assistants such as ChatGPT-4, Gemini Ultra or TelecomGPT (a third-party application built on ChatGPT to provide specialization on the telecommunication sector²). The responses are summarized in Table 3 for the query “*What are the information elements included in the ‘ECN Failure Indication’, and how are they defined?*” presented earlier. As it can be observed, RAG-based systems are expected to provide more useful responses for Telecom-based use cases.

5 DISCUSSION AND FUTURE WORK

This *Editorial Note* explored the use of LLM-based solutions, in particular, Retrieval- Augmented Generation (RAG), for building telecommunication standard assistants. We discussed the different aspects of the conceptual design of the framework as well as the implementation details needed to realize such kind of systems. The solution is simple and practical, and thanks to today’s open-source libraries, the development of such tools becomes very accessible.

The use of RAG brings many advantages. First, it enhances the standalone LLMs by grounding the responses to factual information. Second, RAG allows the LLMs to elaborate answers based on multiple documents or sources of knowledge (e.g., mailing lists, online resources, etc.). Last but not least, the chat close loop allows the iterative refinement of the response to satisfy the user’s needs. However, the main disadvantage associated with generative models is hallucination. Although the RAG approach minimizes this phenomenon, the responses generated by LLMs are not free from factual errors when, for example, the information provided in the retrieval phase is insufficient. Unfortunately, this issue is difficult to detect and fix in this kind of system.

The application of RAG to telecom standard documents also brings many challenges to be addressed in future work. Some forms of data representation are difficult to process by the system, such as tables, figures, and diagrams. The system may struggle with acronyms due to the lack of context, which may lead to hallucinations. Moreover, the cross-references across different sections (or even documents) very common in standard documents are difficult to follow by the system, limiting the completeness of the responses. Finally, the retrieval techniques used in RAG systems are based on semantic similarity. However, this approach needs to be extended to consider other forms of information representation such as pictures, diagrams, or even network logs.

In conclusion, while this study has addressed several key aspects of RAG-based systems in telecommunications, there are many open issues and more research efforts are required in this area. We believe that this *Editorial Note* not only opens the door to new research avenues but also has the potential to increase the interest of the academic and industrial communities in this promising approach. The integration of RAG-based systems holds significant promise for enhancing various applications within the telecommunications field, offering substantial value and opportunities.

²Available in <https://chat.openai.com/g/g-eSwyzwwDV-telecom-gpt>.

Table 3: What are the information elements included in the “ECN Failure Indication”, and how are they defined?

ChatGPT 4's response	<p>The “ECN Failure Indication” is a term that is typically associated with network protocols, specifically with mechanisms that deal with congestion control. ECN stands for Explicit Congestion Notification. This feature enables network nodes to notify each other about network congestion without dropping packets. However, the specific term “ECN Failure Indication” isn’t standard, and you may be referring to situations where ECN signaling fails or is not properly communicated between network entities. Here’s a detailed look at the typical elements related to ECN, which might be relevant to a failure scenario:</p> <ol style="list-style-type: none"> (1) ECN-Capable Transport (ECT) - This is a codepoint in the IP header that indicates that the end-points of the TCP connection are ECN-capable, meaning they can both send and receive ECN signals. There are two ECT codepoints, ECT(0) and ECT(1), which serve slightly different purposes but essentially signal ECN capabilities. (2) Congestion Experienced (CE) - This is another codepoint in the IP header used by routers experiencing congestion to indicate this status to the end-points of the transmission. When a packet marked with an ECT codepoint encounters congestion, routers that support ECN can mark these packets as CE instead of dropping them. (3) ECN Echo (ECE) - This is a flag in the TCP header used to indicate that a segment with the CE codepoint has been received. The receiving end uses this flag to notify the sender about the congestion. (4) Congestion Window Reduced (CWR) - This flag is set by the sending end to signal to the receiving end that it has received the ECE flag and has reduced its congestion window size accordingly. This is an acknowledgment that the sender is responding to the congestion. <p>When discussing “ECN Failure Indication”, it might relate to any anomalies or failures in the signaling process among these elements. Here are a few scenarios where ECN could “fail”:</p> <ul style="list-style-type: none"> • Unsupported ECN: If any part of the path between sender and receiver does not support ECN, then the ECN signals (ECT, CE) may not be handled or propagated correctly. • Incorrect ECN Implementation: Bugs or incorrect implementations in routers or end hosts can lead to incorrect ECN behavior, such as failing to set CE flags despite congestion or not reacting to ECE flags. • ECN Negotiation Failure: During the TCP handshake, both sides must agree to use ECN. If there is a failure in this negotiation, ECN will not be used, even if both endpoints support it. <p>Understanding specific ECN failure modes would depend on the context in which ECN is being used (like in specific network architectures or for certain types of traffic) and the specific tools and methods used for diagnosing such failures.</p>
Gemini Ultra's response	<p>Unfortunately, there isn’t a universally defined “ECN Failure Indication” with a set list of information elements. “ECN” can refer to different things depending on the context, and the specific details of a failure indication would depend on the particular system or protocol involved. Here’s a breakdown of the possibilities:</p> <ul style="list-style-type: none"> • IP Networking: In the context of Internet Protocol (IP) networking, ECN (Explicit Congestion Notification) is a mechanism for routers to signal congestion to senders. It’s unlikely there would be a separate “ECN Failure Indication”. Issues with ECN functionality might be reported through other mechanisms within the specific routing protocol. • Other Fields: ECN can also stand for Engineering Change Notice or Entry Change Notice in various fields. A failure related to an ECN would likely be specific to the system or process that uses those terms and wouldn’t have a standardized format. <p>Here are some ways to get more information:</p> <ul style="list-style-type: none"> • Context Clues: If you encountered the term “ECN Failure Indication” somewhere specific, look for surrounding information that might indicate the context (e.g., manuals, software documentation). • System Identification: If you know the system or protocol where you came across this term, searching for its documentation might reveal details about error reporting mechanisms.
TelecomGPT response	<p>If you can provide more context about where you encountered “ECN Failure Indication”, I might be able to offer more specific guidance.</p> <p>The “ECN Failure Indication” message in the context of telecommunications and protocols isn’t explicitly defined in the documents I have available, such as the SMPP specifications and various telecom-related white papers and presentations. It appears you might be referring to a specific protocol message or error indication that might be part of a broader telecommunications standard or proprietary system not covered in these documents. If the “ECN Failure Indication” you are asking about is related to a specific application, protocol, or vendor-specific implementation, it would be helpful to have more context or details about the system in which this term is used. If it is part of a standard not covered in the documents or discussions I have access to, you may need to consult the specific documentation or standards applicable to that protocol or system. If you can provide more details or clarify the specific protocol or context in which you encountered “ECN Failure Indication”, I can assist you better. Alternatively, if this is related to a standard or system, the relevant documentation or technical support resources for that system may be the best source for detailed information.</p>

REFERENCES

- [1] 3rd Generation Partnership Project (3GPP). 2021. *IP Multimedia Subsystem (IMS) Application Level Gateway (IMS-ALG) - IMS Access Gateway (IMS-AGW) interface*. Technical Specification 23.334. 3rd Generation Partnership Project (3GPP).
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [3] Jacob Devlin et al. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [4] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. In *International conference on machine learning*. PMLR, 3929–3938.
- [5] Katharina Jeblick et al. 2023. ChatGPT makes medicine easy to swallow: an exploratory case study on simplified radiology reports. *European radiology* (2023), 1–9.
- [6] Jared Kaplan et al. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).
- [7] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).
- [8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
- [9] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems* 35 (2022), 3843–3857.
- [10] Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfahl, et al. 2022. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138* (2022).
- [11] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085* (2022).
- [12] Oguzhan Topsakal and Tahir Cetin Akinci. 2023. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, Vol. 1. 1050–1056.
- [13] Jason Wei et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.
- [14] Jason Wei et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).
- [15] Shijie Wu et al. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564* (2023).