Using Amazon Comprehend Medical and LLMs for healthcare and life sciences

AWS Prescriptive Guidance

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Prescriptive Guidance: Using Amazon Comprehend Medical and LLMs for healthcare and life sciences

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Introduction **1**
Overview 1
Intended audience 2
Objectives 2
Technical approaches **4**
Using Amazon Comprehend Medical 4
Amazon Comprehend Medical capabilities 5
Use cases for Amazon Comprehend Medical 6
Combining Amazon Comprehend Medical with LLMs 7
RAG-based architecture with Amazon Comprehend Medical 8
Use cases for using Amazon Comprehend Medical in a RAG workflow 9
Best practices for using Amazon Comprehend Medical in a RAG workflow 10
Prompt engineering for Amazon Comprehend Medical context 11
Using LLMs 20
Use cases for an LLM 21
Customization approaches 21
Choosing an LLM 24
Fine-tuning LLMs 27
Estimating costs and return on investment 28
Choosing a fine-tuning strategy 29
Building a fine-tuning dataset 31
Fine-tuning the model 32
Monitoring fine-tuned models 33
Choosing an approach **34**
Business maturity considerations 35
Evaluating LLMs **37**
Training and test data for medical NLP tasks 37
Metrics for medical NLP tasks 38
FAQ **40**
How do I choose between Amazon Comprehend Medical and an LLM? 40
How can I provide Amazon Comprehend Medical results to an LLM? 40
What are some best practices when using Amazon Comprehend Medical with LLMs? 40
Should I use a pretrained medical LLM or fine-tune a general LLM for my healthcare use case? 41

How do I evaluate the performance of LLMs for medical NLP tasks? 41
What are the trade-offs between high-complexity and low-complexity LLM solutions? 41
Next steps 42
AWS resources 42
Other resources 43
Contributors 44
Authoring 44
Reviewing 44
Technical writing 44
Document history 45

Using Amazon Comprehend Medical and LLMs for healthcare and life sciences

Joe King, Ross Claytor, Ankith Ede, Jillian Forde, Clement Perrot, Rajesh Sitaraman, and Shivesh Ummat, Amazon Web Services

Overview

The ever-increasing volume of medical data and the need for efficient and accurate processing have driven the adoption of natural language processing (NLP) with artificial intelligence and machine learning (AI/ML) technologies. Pretrained classifier models and large language models (LLMs) have emerged as powerful tools for various medical NLP tasks, including clinical question-answering, report summarization, and insight generation. However, the healthcare and life science domain presents unique challenges due to the complexity of medical terminology, domain-specific knowledge, and regulatory requirements. Effectively using pretrained classifiers or LLMs in this domain requires a well-designed approach that combines the strengths of these models with domain-specific resources and techniques.

Industry practices in healthcare and life science have traditionally relied on rule-based systems, manual coding, and expert review processes. These systems and processes are time-consuming and error-prone. The integration of AI and NLP technologies, such as Amazon Comprehend Medical and the foundation models in Amazon Bedrock, offers efficient and scalable solutions for processing medical data while improving accuracy and consistency.

This guide explores the use of Amazon Comprehend Medical and LLMs for intelligent automation in the healthcare industry. It outlines best practices, challenges, and practical approaches to streamlining medical coding, patient information extraction, and record summarization processes. By using the capabilities of Amazon Comprehend Medical and LLMs, healthcare organizations can unlock new levels of operational efficiency, reduce costs, and potentially improve patient care.

The guide details the unique considerations of the healthcare domain, such as understanding medical terminology, using domain-specific LLMs, and addressing the limitations of AI/ML systems. It provides a comprehensive decision path for healthcare IT managers, architects, and technical leads to assess organizational readiness, evaluate implementation options, and use the appropriate AWS services and tools for successful automation.

By following the guidelines and best practices outlined in this guide, healthcare organizations can harness the power of AI/ML technologies while navigating the complexities of the medical domain. This approach supports compliance with ethical and regulatory guidelines and promotes the responsible use of AI systems in healthcare. It is designed to generate insights that are accurate and private.

Intended audience

This guide is intended for technology stakeholders, architects, technical leads, and decision makers who want to implement AI-powered natural language processing solutions for medical data analysis and automation.

Objectives

Healthcare and life science organizations can meet multiple business goals by using Amazon Comprehend Medical and LLMs. These outcomes commonly include increasing operational efficiency, reducing costs, and improving patient care. This section outlines key business objectives and the associated benefits of implementing the strategies and best practices outlined in this guide.

The following are the some of the objectives that organizations can achieve by implementing the guidelines and best practices in this guide:

- **Reduce the development time** – This guide's ultimate goal is to reduce development time with cost, decrease technical debt, and mitigate potential project failure from POC. By understanding key AI/ML services, such as Amazon Comprehend Medical, and the advantages and limitations of LLM usage for healthcare tasks, businesses can achieve faster time to market and increase their velocity in meeting business objectives.
- **Extract information to automate medical coding tasks** – After patient visits, coding specialist and providers can extract insights from medical text, such as subjective, objective, assessment, and plan (SOAP) notes. This can reduce manual documentation efforts and help the provider focus on the patient's needs. By combining the entity recognition capabilities of Amazon Comprehend Medical with LLMs, organizations can extract relevant medical information from patient records, clinical notes, and other healthcare data sources. This can minimize human errors and promote consistent practices.
- **Summarize patient records and clinical documentation** – Automated summarization of patient history, treatment plans, and medical results can save valuable time for healthcare providers.

LLMs can help generate comprehensive and structured clinical documentation. You can get additional context with Amazon Comprehend Medical, use a medical domain LLM, or fine-tune an LLM with medical data. These approaches can help provide accurate summaries and make sure that documentation adheres to compliance requirements and standards.

- **Support clinical decisions and patient care** – By using ontology linking in Amazon Comprehend Medical and by using LLMs, providers can answer medical questions or seek recommendations addressing patient care. This empowers healthcare professionals to make informed decisions that improve patient outcomes and reduce the risk of medical errors.

Generative AI and NLP approaches for healthcare and life sciences

Natural language processing (NLP) is a machine learning technology that gives computers the ability to interpret, manipulate, and comprehend human language. Healthcare and life science organizations have large volumes of data from patient records. They can use NLP software to automatically process this data. For example, they can combine NLP with generative AI to streamline medical coding, extract patient information, and summarize records.

Depending on the NLP task that you want to perform, different architectures might be best suited for your use case. This guide addresses the following generative AI and NLP options for healthcare and life science applications on AWS:

- Using Amazon Comprehend Medical – Learn about how to use Amazon Comprehend Medical independently, without integrating it with a large language model (LLM).
- Combining Amazon Comprehend Medical with LLMs – Learn about how to combine Amazon Comprehend Medical with an LLM in a Retrieval Augment Generation (RAG) architecture.
- Using LLMs – Learn about how to use an LLM for healthcare and life science applications, either by using a fine-tuned LLM or a RAG architecture.

Using Amazon Comprehend Medical

Amazon Comprehend Medical is an AWS service that detects and returns useful information in unstructured clinical text such as physician's notes, discharge summaries, test results, and case notes. It uses natural language processing (NLP) models to detect entities. *Entities* are textual references to medical information, such as medical conditions, medications, or protected health information (PHI).

> **Important**
>
> Amazon Comprehend Medical is not a substitute for professional medical advice, diagnosis, or treatment. Amazon Comprehend Medical provides confidence scores that indicate the level of confidence in the accuracy of the detected entities. Identify the right confidence threshold for your use case, and use high confidence thresholds in situations that require high accuracy. In certain use cases, results should be reviewed and verified by appropriately

trained human reviewers. For example, Amazon Comprehend Medical should only be used in patient care scenarios after review for accuracy and sound medical judgment by trained medical professionals.

You can access Amazon Comprehend Medical through the AWS Management Console, the AWS Command Line Interface (AWS CLI), or through the AWS SDKs. The AWS SDKs are available for various programming languages and platforms, such as Java, Python, Ruby, .NET, iOS, and Android. You can use the SDKs to programmatically access Amazon Comprehend Medical from your client application.

This section reviews the main capabilities of Amazon Comprehend Medical. It also discusses the advantages of using this service compared to a large language model (LLM).

Amazon Comprehend Medical capabilities

Amazon Comprehend Medical offers APIs for near real-time and batch inference. These APIs can ingest medical text and provide results for medical NLP tasks by using medical entity recognition and identifying entity relationships. You can perform analysis both on single files or as a batch analysis on multiple files stored in an Amazon Simple Storage Service (Amazon S3) bucket. Amazon Comprehend Medical offers the following text analysis API operations for synchronous entity detection:

- Detect entities – Detects general medical categories such as anatomy, medical condition, PHI category, procedures, and time expressions.
- Detect PHI – Detects specific entities such as age, date, name, and similar personal information.

Amazon Comprehend Medical also includes multiple API operations that you can use to perform batch text analysis on clinical documents. To learn more about how to use these API operations, see Text analysis batch APIs.

Use Amazon Comprehend Medical to detect entities in clinical text and link those entities to concepts in standardized medical ontologies, including the RxNorm, ICD-10-CM, and SNOMED CT knowledge bases. You can perform analysis both on single files or as a batch analysis on large documents or multiple files stored in an Amazon S3 bucket. Amazon Comprehend Medical offers the following ontology linking API operations:

- InferICD10CM – The **InferICD10CM** operation detects potential medical conditions and links them to codes from the 2019 version of the International Classification of Diseases, 10th Revision, Clinical Modification (ICD-10-CM). For each potential medical condition detected, Amazon Comprehend Medical lists the matching ICD-10-CM codes and descriptions. Listed medical conditions in the results include a confidence score, which indicates the confidence that Amazon Comprehend Medical has in the accuracy of the entities to the matched concepts in the results.
- InferRxNorm – The **InferRxNorm** operation identifies medications that are listed in a patient record as entities. It links entities to concept identifiers (RxCUI) from the RxNorm database from the National Library of Medicine. Each RxCUI is unique for different strengths and dose forms. Listed medications in the results include a confidence score, which indicates the confidence that Amazon Comprehend Medical has in the accuracy of the entities matched to the concepts from the RxNorm knowledge base. Amazon Comprehend Medical lists the top RxCUIs that potentially match for each medication that it detects in descending order based on confidence score.
- InferSNOMEDCT – The **InferSNOMEDCT** operation identifies possible medical concepts as entities and links them to codes from the 2021-03 version of the Systematized Nomenclature of Medicine, Clinical Terms (SNOMED CT). SNOMED CT provides a comprehensive vocabulary of medical concepts, including medical conditions and anatomy, as well as medical tests, treatments, and procedures. For each matched concept ID, Amazon Comprehend Medical returns the top five medical concepts, each with a confidence score and contextual information such as traits and attributes. The SNOMED CT concept IDs can then be used to structure patient clinical data for medical coding, reporting, or clinical analytics when used with the SNOMED CT polyhierarchy.

For more information, see Text analysis APIs and Ontology Linking APIs in the Amazon Comprehend Medical documentation.

Use cases for Amazon Comprehend Medical

As a standalone service, Amazon Comprehend Medical might address your organization's use case. Amazon Comprehend Medical can perform tasks such as the following:

- Help with medical coding in patient records
- Detect protected health information (PHI) data
- Validating medication, including attributes such as dosage, frequency, and form

Amazon Comprehend Medical results are digestible for the majority of medical practices. However, you might need to consider alternatives if you have limitations such as the following:

- **Different entity definitions** – For example, your definition of FREQUENCY of a medication entity might differ. For frequency, Amazon Comprehend Medical predicts *as needed*, but your organization might use the term *pro re nata (PRN)*.
- **Overwhelming quantity of results** – For example, patient notes frequently contain multiple symptoms and keywords that map to multiple ICD-10-CM codes. However, several of the keywords are not applicable for diagnosis. In this case, the provider has to evaluate numerous ICD-10-CM entities and their confidence scores, which requires manual processing time.
- **Custom entities or NLP tasks** – For example, providers might want to extract PRN evidence, such as *take as needed for pain*. Because this isn't available through Amazon Comprehend Medical, a different AI/ML model is warranted. A different AI/ML solution is required if the NLP task is outside of entity recognition, such as summarization, question-answering, and sentiment analysis.

Combining Amazon Comprehend Medical with large language models

A 2024 study by NEJM AI showed that using an LLM, with zero-shot prompting, for medical coding tasks generally leads to poor performance. Using Amazon Comprehend Medical with an LLM can help mitigate these performance issues. Amazon Comprehend Medical results are helpful context for an LLM that is performing NLP tasks. For example, providing context from Amazon Comprehend Medical to the large language model can help you:

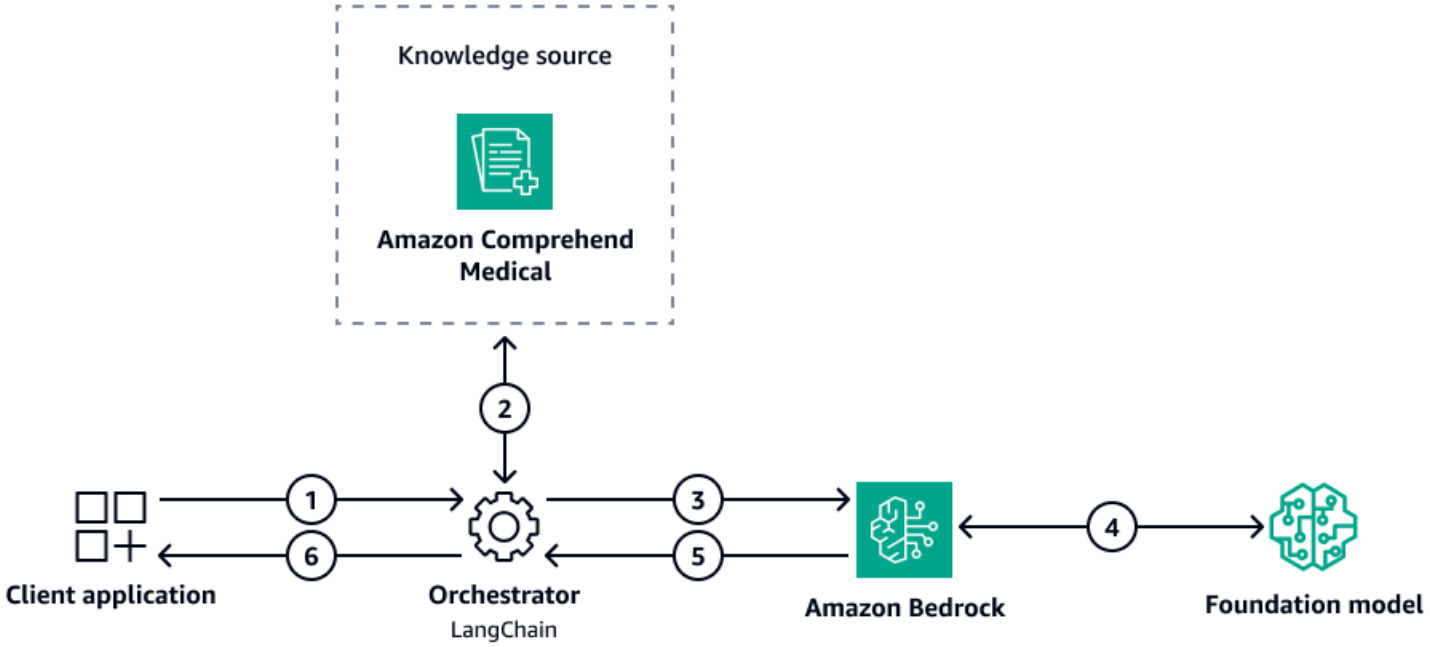
- Enhance the accuracy of entity selections by using the initial results from Amazon Comprehend Medical as context for the LLM
- Implement custom entity recognition, summarization, question-answering, and additional use cases

This section describes how you can combine Amazon Comprehend Medical with an LLM by using a Retrieval Augmented Generation (RAG) approach. *Retrieval Augmented Generation (RAG)* is a generative AI technology in which an LLM references an authoritative data source that is outside of its training data sources before generating a response. For more information, see What is RAG.

To illustrate this approach, this section uses the example of medical (diagnosis) coding related to ICD-10-CM. It includes a sample architecture and prompt engineering templates to help accelerate your innovation. It also includes best practices for using Amazon Comprehend Medical within a RAG workflow.

RAG-based architecture with Amazon Comprehend Medical

The following diagram illustrates a RAG approach for identifying ICD-10-CM diagnosis codes from patient notes. It uses Amazon Comprehend Medical as a knowledge source. In a RAG approach, the retrieval method commonly retrieves information from a vector database containing applicable knowledge. Instead of a vector database, this architecture uses Amazon Comprehend Medical for the retrieval task. The orchestrator sends the patient note information to Amazon Comprehend Medical and retrieves the ICD-10-CM code information. The orchestrator sends this context to the downstream foundation model (LLM), through Amazon Bedrock. The LLM generates a response by using the ICD-10-CM code information, and that response is sent back to the client application.



The diagram shows the following RAG workflow:

1. The client application sends the patient notes as a query to the orchestrator. An example of these patient notes might be "The patient is a 71-year-old female patient of Dr. X. The patient presented to the emergency room last evening with approximately 7-day to 8-day history of abdominal pain, which has been persistent. She has had no definite fevers or chills and no history of jaundice. The patient denies any significant recent weight loss."

2. The orchestrator uses Amazon Comprehend Medical to retrieve ICD-10-CM codes relevant to the medical information in the query. It uses the **InferICD10CM** API to extract and infer the ICD-10-CM codes from the patient notes.
3. The orchestrator constructs a prompt that includes the prompt template, the original query, and the ICD-10-CM codes retrieved from Amazon Comprehend Medical. It sends this enhanced context to Amazon Bedrock.
4. Amazon Bedrock processes the input and uses a foundation model to generate a response that includes the ICD-10-CM codes and their corresponding evidence from the query. The generated response includes the identified ICD-10-CM codes and evidence from the patient notes that supports each code. The following is a sample response:

```
<response>
<icd10>
<code>R10.9</code>
<evidence>history of abdominal pain</evidence>
</icd10>
<icd10>
<code>R10.30</code>
<evidence>history of abdominal pain</evidence>
</icd10>
</response>
```

5. Amazon Bedrock sends the generated response to the orchestrator.
6. The orchestrator sends the response back to the client application, where the user can review the response.

Use cases for using Amazon Comprehend Medical in a RAG workflow

Amazon Comprehend Medical can perform specific NLP tasks. For more information, see Use cases for Amazon Comprehend Medical.

You might want to integrate Amazon Comprehend Medical into a RAG workflow for advanced use cases, such as the following:

- Generate detailed clinical summaries by combining extracted medical entities with contextual information from patient records
- Automate medical coding for complex cases by using extracted entities with ontology-linked information for code assignment

- Automate the creation of structured clinical notes from unstructured text by using extracted medical entities
- Analyze medication side effects based on extracted medication names and attributes
- Develop intelligent clinical support systems that combine extracted medical information with up-to-date research and guidelines

Best practices for using Amazon Comprehend Medical in a RAG workflow

When integrating Amazon Comprehend Medical results into a prompt for an LLM, it's essential to follow best practices. This can improve performance and accuracy. The following are key recommendations:

- **Understand Amazon Comprehend Medical confidence scores** – Amazon Comprehend Medical provides confidence scores for each detected entity and ontology linking. It's crucial to understand the meaning of these scores and establish appropriate thresholds for your specific use case. Confidence scores help filter out low-confidence entities, reducing noise and improving the quality of the LLM's input.
- **Use confidence scores in prompt engineering** – When crafting prompts for the LLM, consider incorporating Amazon Comprehend Medical confidence scores as additional context. This helps the LLM prioritize or weigh entities based on their confidence levels, potentially improving the quality of the output.
- **Evaluate Amazon Comprehend Medical results with ground truth data** – *Ground truth data* is information that is known to be true. It can be used to validate that an AI/ML application is producing accurate results. Before integrating Amazon Comprehend Medical results into your LLM workflow, evaluate the service's performance on a representative sample of your data. Compare the results with ground truth annotations to identify potential discrepancies or areas for improvement. This evaluation helps you understand the strengths and limitations of Amazon Comprehend Medical for your use case.
- **Strategically select relevant information** – Amazon Comprehend Medical can provide a large amount of information, but not all of it may be relevant to your task. Carefully select the entities, attributes, and metadata that are most relevant to your use case. Providing too much irrelevant information to the LLM can introduce noise and potentially decrease performance.
- **Align entity definitions** – Ensure that the definitions of entities and attributes used by Amazon Comprehend Medical align with your interpretation. If there are discrepancies, consider

providing additional context or clarification to the LLM to bridge the gap between the Amazon Comprehend Medical output and your requirements. If Amazon Comprehend Medical entity doesn't meet your expectations, you can implement custom entity detection by including additional instructions (and possible examples) within the prompt.

- **Provide domain-specific knowledge** – While Amazon Comprehend Medical provides valuable medical information, it might not capture all the nuances of your specific domain. Consider supplementing Amazon Comprehend Medical results with additional domain-specific knowledge sources, such as ontologies, terminologies, or expert-curated datasets. This provides more comprehensive context to the LLM.
- **Adhere to ethical and regulatory guidelines** – When dealing with medical data, it's important to adhere to ethical principles and regulatory guidelines, such as those related to data privacy, security, and responsible use of AI systems in healthcare. Make sure that your implementation complies with relevant laws and industry best practices.

By following these best practices, AI/ML practitioners can effectively use the strengths of both Amazon Comprehend Medical and LLMs. For medical NLP tasks, these best practices help mitigate potential risks and can improve performance.

Prompt engineering for Amazon Comprehend Medical context

Prompt engineering is the process of designing and refining prompts to guide a generative AI solution to generate desired outputs. You choose the most appropriate formats, phrases, words, and symbols that guide the AI to interact with your users more meaningfully.

Depending on the API operation you perform, Amazon Comprehend Medical returns the detected entities, ontology codes and descriptions, and confidence scores. These results become context within the prompt when your solution invokes the target LLM. You must engineer the prompt to present the context within the prompt template.

> **Note**
>
> The example prompts in this section follow Anthropic guidance. If you're using a different LLM provider, follow the recommendations from that provider.

In general, you insert both the original medical text and the Amazon Comprehend Medical results into the prompt. The following is a common prompt structure:

```
<medical_text>
medical text
</medical_text>

<comprehend_medical_text_results>
comprehend medical text results
</comprehend_medical_text_results>

<prompt_instructions>
prompt instructions
</prompt_instructions>
```

This section provides strategies for including Amazon Comprehend Medical results as prompt context for the following common medical NLP tasks:

- Filter Amazon Comprehend Medical results
- Extend medical NLP tasks with Amazon Comprehend Medical
- Apply guardrails with Amazon Comprehend Medical

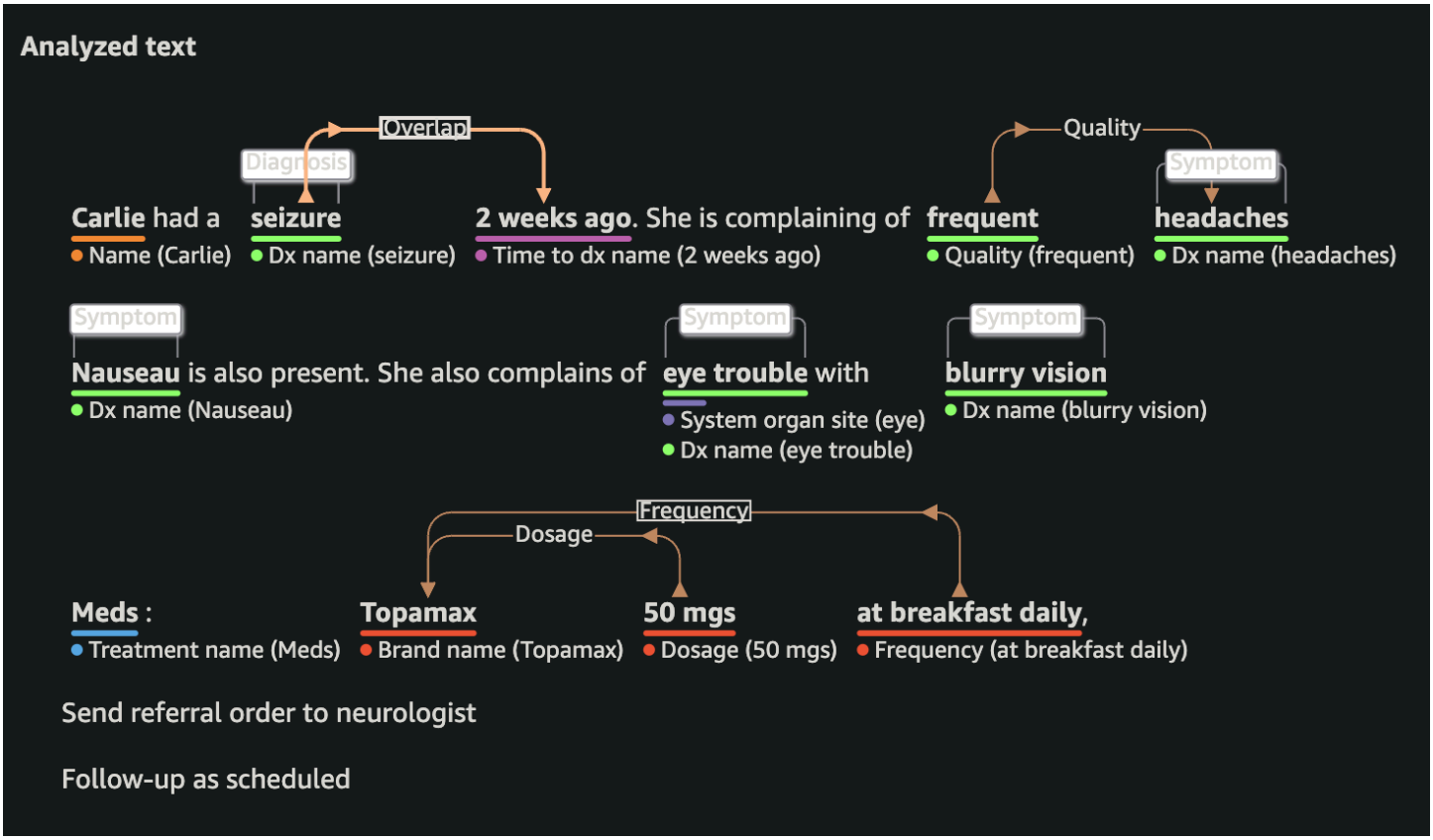
Filter Amazon Comprehend Medical results

Amazon Comprehend Medical typically provides a large amount of information. You might want to reduce the number of results that the medical professional must review. In this case, you can use an LLM to filter these results. Amazon Comprehend Medical entities include a confidence score that you can use as a filtering mechanism when designing the prompt.

The following is an example patient note:

```
Carlie had a seizure 2 weeks ago. She is complaining of frequent headaches
Nausea is also present. She also complains of eye trouble with blurry vision
Meds : Topamax 50 mgs at breakfast daily,
Send referral order to neurologist
Follow-up as scheduled
```

In this patient note, Amazon Comprehend Medical detects the following entities.



The entities link to the following ICD-10-CM codes for seizure and headaches.

Category	ICD-10-CM code	ICD-10-CM description	Confidence score
Seizure	R56.9	Unspecified convulsions	0.8348
Seizure	G40.909	Epilepsy, unspecified, not intractable, without status epilepticus	0.5424
Seizure	R56.00	Simple febrile convulsions	0.4937
Seizure	G40.09	Other seizures	0.4397

Seizure	G40.409	Other generalized epilepsy and epileptic syndromes, not intractable, without status epilepticus	0.4138
Headaches	R51	Headache	0.4067
Headaches	R51.9	Headache, unspecified	0.3844
Headaches	G44.52	New daily persistent headache (NDPH)	0.3005
Headaches	G44	Other headache syndrome	0.2670
Headaches	G44.8	Other specified headache syndromes	0.2542

You can pass ICD-10-CM codes into the prompt to increase LLM precision. To reduce noise, you can filter the ICD-10-CM codes by using the confidence score included in the Amazon Comprehend Medical results. The following is an example prompt that includes only ICD-10-CM codes that have a confidence score higher than 0.4:

```
<patient_note>
Carlie had a seizure 2 weeks ago. She is complaining of frequent headaches
Nausea is also present. She also complains of eye trouble with blurry vision
Meds : Topamax 50 mgs at breakfast daily,
Send referral order to neurologist
Follow-up as scheduled
</patient_note>

<comprehend_medical_results>
<icd-10>
  <entity>
    <text>seizure</text>
    <code>
      <description>Unspecified convulsions</description>
```

```
      <code_value>R56.9</code_value>
      <score>0.8347607851028442</score>
    </code>
    <code>
      <description>Epilepsy, unspecified, not intractable, without status epilepticus</
description>
      <code_value>G40.909</code_value>
      <score>0.542376697063446</score>
    </code>
    <code>
      <description>Other seizures</description>
      <code_value>G40.89</code_value>
      <score>0.43966275453567505</score>
    </code>
    <code>
      <description>Other generalized epilepsy and epileptic syndromes, not intractable,
 without status epilepticus</description>
      <code_value>G40.409</code_value>
      <score>0.41382506489753723</score>
    </code>
  </entity>
  <entity>
    <text>headaches</text>
    <code>
      <description>Headache</description>
      <code_value>R51</code_value>
      <score>0.4066613018512726</score>
    </code>
  </entity>
  <entity>
    <text>Nausea</text>
    <code>
      <description>Nausea</description>
      <code_value>R11.0</code_value>
      <score>0.6460834741592407</score>
    </code>
  </entity>
  <entity>
    <text>eye trouble</text>
    <code>
      <description>Unspecified disorder of eye and adnexa</description>
      <code_value>H57.9</code_value>
      <score>0.6780954599380493</score>
```

```
    </code>
    <code>
      <description>Unspecified visual disturbance</description>
      <code_value>H53.9</code_value>
      <score>0.5871203541755676</score>
    </code>
    <code>
      <description>Unspecified disorder of binocular vision</description>
      <code_value>H53.30</code_value>
      <score>0.5539672374725342</score>
    </code>
  </entity>
  <entity>
    <text>blurry vision</text>
    <code>
      <description>Other visual disturbances</description>
      <code_value>H53.8</code_value>
      <score>0.9001834392547607</score>
    </code>
  </entity>
</icd-10>
</comprehend_medical_results>

<prompt>
Given the patient note and Amazon Comprehend Medical ICD-10-CM code results above,
 please select the most relevant ICD-10-CM diagnosis codes for the patient.
For each selected code, provide a brief explanation of why it is relevant based on the
 information in the patient note.
</prompt>
```

Extend medical NLP tasks with Amazon Comprehend Medical

When processing medical text, context from Amazon Comprehend Medical can help the LLM to select better tokens. In this example, you want to match diagnosis symptoms to medications. You also want to find text that relates to medical tests, such as terms that relate to a blood panel test. You can use Amazon Comprehend Medical to detect the entities and the medication names. In this case, you would use the DetectEntitiesV2 and InferRxNorm APIs for Amazon Comprehend Medical.

The following is an example patient note:

```
Carlie had a seizure 2 weeks ago. She is complaining of increased frequent headaches
Given lyme disease symptoms such as muscle ache and stiff neck will order prescription.
```

```
Meds : Topamax 50 mgs at breakfast daily. Amoxicillan 25 mg by mouth twice a day
Place MRI radiology order at RadNet
```

To focus on the diagnosis code, only the entities related to `MEDICAL_CONDITION` with type `DX_NAME` are used in the prompt. Other metadata is excluded due to irrelevance. For medication entities, the medication name along with extracted attributes is included. Other medication entity metadata from Amazon Comprehend Medical is excluded due to irrelevance. The following is an example prompt that uses filtered Amazon Comprehend Medical results. The prompt focuses on `MEDICAL_CONDITION` entities that have the `DX_NAME` type. This prompt is designed to more precisely link diagnosis codes with medication and more precisely extract medical order tests:

```
<patient_note>
Carlie had a seizure 2 weeks ago. She is complaining of increased freqeunt headaches
Given lyme disease symptoms such as muscle ache and stiff neck will order
 prescription.
Meds : Topamax 50 mgs at breakfast daily. Amoxicillan 25 mg by mouth twice a day
Place MRI radiology order at RadNet
</patient_note>

<detect_entity_results>
<entity>
    <text>seizure</text>
    <category>MEDICAL_CONDITION</category>
    <type>DX_NAME</type>
</entity>
<entity>
    <text>headaches</text>
    <category>MEDICAL_CONDITION</category>
    <type>DX_NAME</type>
</entity>
<entity>
    <text>lyme disease</text>
    <category>MEDICAL_CONDITION</category>
    <type>DX_NAME</type>
</entity>
<entity>
    <text>muscle ache</text>
    <category>MEDICAL_CONDITION</category>
    <type>DX_NAME</type>
</entity>
<entity>
    <text>stiff neck</text>
```

```
    <category>MEDICAL_CONDITION</category>
    <type>DX_NAME</type>
</entity>
</detect_entity_results>

<rx_results>
<entity>
    <text>Topamax</text>
    <category>MEDICATION</category>
    <type>BRAND_NAME</type>
    <attributes>
        <attribute>
            <type>FREQUENCY</type>
            <text>at breakfast daily</text>
        </attribute>
        <attribute>
            <type>DOSAGE</type>
            <text>50 mgs</text>
        </attribute>
        <attribute>
            <type>ROUTE_OR_MODE</type>
            <text>by mouth</text>
        </attribute>
    </attributes>
</entity>
<entity>
    <text>Amoxicillan</text>
    <category>MEDICATION</category>
    <type>GENERIC_NAME</type>
    <attributes>
        <attribute>
            <type>ROUTE_OR_MODE</type>
            <text>by mouth</text>
        </attribute>
        <attribute>
            <type>DOSAGE</type>
            <text>25 mg</text>
        </attribute>
        <attribute>
            <type>FREQUENCY</type>
            <text>twice a day</text>
        </attribute>
    </attributes>
```

```
</entity>
</rx_results>

<prompt>
Based on the patient note and the detected entities, can you please:
1. Link the diagnosis symptoms with the medications prescribed.
Provide your reasoning for the linkages.
2. Extract any entities related to medical order tests mentioned in the note.
</prompt>
```

Apply guardrails with Amazon Comprehend Medical

You can use an LLM and Amazon Comprehend Medical to create guardrails before the generated response is used. You can run this workflow on either unmodified or post-processed medical text. Use cases include addressing protected health information (PHI), detecting hallucinations, or implementing custom policies for publishing results. For example, you can use context from Amazon Comprehend Medical to identify PHI data and then use the LLM to remove that PHI data.

The following is an example of information from a patient record that includes PHI:

```
Patient name: John Doe
Patient SSN: 123-34-5678
Patient DOB: 01/01/2024
Patient address: 123 Main St, Anytown USA
Exam details: good health. Pulse is 60 bpm. needs to work on diet with BMI of 190
```

The following is an example prompt that includes the Amazon Comprehend Medical results as context:

```
<original_text>
Patient name: John Doe
Patient SSN: 123-34-5678 Patient DOB: 01/01/2024
Patient address: 123 Main St, Anytown USA
Exam details: good health. Pulse is 60 bpm. needs to work on diet with BMI of 190
</original_text>

<comprehend_medical_phi_entities>
<entity>
  <text>John Doe</text>
  <category>PROTECTED_HEALTH_INFORMATION</category>
  <score>0.9967944025993347</score>
```

```
  <type>NAME</type>
</entity>
<entity>
  <text>123-34-5678</text>
  <category>PROTECTED_HEALTH_INFORMATION</category>
  <score>0.9998034834861755</score>
  <type>ID</type>
</entity>
<entity>
  <text>01/01/2000</text>
  <category>PROTECTED_HEALTH_INFORMATION</category>
  <score>0.9964448809623718</score>
  <type>DATE</type>
</entity>
</comprehend_medical_phi_entities>

<instructions>
Using the provided original text and the Amazon Comprehend Medical PHI entities
 detected, please analyze the text to determine if it contains any additional protected
 health information (PHI) beyond the entities already identified. If additional PHI is
 found, please list and categorize it. If no additional PHI is found, please state that
 explicitly.
In addition if PHI is found, generate updated text with the PHI removed.
</instructions>
```

Using large language models for healthcare and life science use cases

This describes how you can use large language models (LLMs) for healthcare and life science applications. Some use cases require the use of a large language model for generative AI capabilities. There are advantages and limitations for even the most state-of-the-art LLMs, and the recommendations in this section are designed to help you achieve your target results.

You can use the decision path to determine the appropriate LLM solution for your use case, considering factors such as domain knowledge and available training data. Additionally, this section discusses popular pretrained medical LLMs and best practices for their selection and use. It also discusses the trade-offs between complex, high-performance solutions and simpler, lower-cost approaches.

Use cases for an LLM

Amazon Comprehend Medical can perform specific NLP tasks. For more information, see Use cases for Amazon Comprehend Medical.

The logical and generative AI capabilities of an LLM might be required for the advanced healthcare and life science use cases, such as the following:

- Classifying custom medical entities or text categories
- Answering clinical questions
- Summarizing medical reports
- Generating and detecting insights from medical information

Customization approaches

It's critical to understand how LLMs are implemented. LLMs are commonly trained with billions of parameters, including training data from many domains. This training allows the LLM to address most generalized tasks. However, challenges often arise when domain-specific knowledge is required. Examples of domain knowledge in healthcare and life science are clinic codes, medical terminology, and health information that is required to generate accurate answers. Therefore, using the LLM as is (zero-shot prompting without supplementing domain knowledge) for these use cases likely results in inaccurate results. There are several popular approaches you can use to overcome this challenge: prompt engineering, Retrieval Augmented Generation (RAG), and fine-tuning.

Prompt engineering

Prompt engineering is the process where you guide generative AI solutions to create the desired outputs by adjusting the inputs to the LLM. By crafting precise prompts with relevant context, it's possible to guide the model towards completion of specialized healthcare tasks that require reasoning. Effective prompt engineering can significantly improve model performance for healthcare use cases without requiring model modifications. For more information about prompt engineering, see Implementing advanced prompt engineering with Amazon Bedrock (AWS blog post). Few-shot prompting and chain-of-thought prompting are techniques that you can use in prompt engineering.

Few-shot prompting

Few-shot prompting is a technique where you provide the LLM with a few examples of the desired input-output before asking it to perform a similar task. In healthcare contexts, this approach is particularly valuable for specialized tasks, such as medical entity recognition or clinical note summarization. By including 3–5 high-quality examples in your prompt, you can significantly improve the model's understanding of medical terminology and domain-specific patterns. For an example of few-shot prompting, see Few-shot prompt engineering and fine-tuning for LLMs in Amazon Bedrock (AWS blog post).

For example, when you extract medication dosages from clinical notes, you can provide examples of different notation styles that help the model recognize variations in how healthcare professionals document prescriptions. This approach is especially effective when working with standardized documentation formats or when consistent patterns exist in the data.

Chain-of-thought prompting

Chain-of-thought (CoT) prompting guides the LLM through a step-by-step reasoning process. This makes it valuable for complex medical decision support and diagnostic reasoning tasks. By explicitly instructing the model to "think step by step" when analyzing clinical scenarios, you can improve its ability to follow medical reasoning protocols and reduce diagnostic errors.

This technique excels when clinical reasoning requires multiple logical steps, such as differential diagnosis or treatment planning. However, this approach has limitations when dealing with highly specialized medical knowledge outside the model's training data or when absolute precision is required for critical care decisions.

In these cases, combining CoT with another approach can yield better results. One option is to combine CoT with self-consistency prompting. For more information, see Enhance performance of generative language models with self-consistency prompting on Amazon Bedrock (AWS blog post). Another option is to combine reasoning frameworks, such as ReAct prompting, with RAG. For more information, see Develop advanced generative AI chat-based assistants by using RAG and ReAct prompting (AWS Prescriptive Guidance).

Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a generative AI technology in which an LLM references an authoritative data source that is outside of its training data sources before generating a response. A RAG system can retrieve medical ontology information (such as international classifications of

diseases, national drug files, and medical subject headings) from a knowledge source. This provides additional context to the LLM to support the medical NLP task.

As discussed in the Combining Amazon Comprehend Medical with large language models section, you can use a RAG approach to retrieve context from Amazon Comprehend Medical. Other common knowledge sources include medical domain data that is stored in a database service, such as Amazon OpenSearch Service, Amazon Kendra, or Amazon Aurora. Extracting information from these knowledge sources can affect retrieval performance, especially with semantic queries that use a vector database.

Another option for storing and retrieving domain-specific knowledge is by using Amazon Q Business in your RAG workflow. Amazon Q Business can index internal document repositories or public-facing web sites (such as CMS.gov for ICD-10 data). Amazon Q Business can then extract relevant information from these sources before passing your query to the LLM.

There are multiple ways to build a custom RAG workflow. For example, there are many ways to retrieve data from a knowledge source. For simplicity, we recommend the common retrieval approach of using a vector database, such as Amazon OpenSearch Service, to store knowledge as embeddings. This requires that you use an embedding model, such as a sentence transformer, to generate embeddings for the query and for the knowledge stored in the vector database.

For more information about fully managed and custom RAG approaches, see Retrieval Augmented Generation options and architectures on AWS.

Fine-tuning

Fine-tuning an existing model involves taking an LLM, such as an Amazon Titan, Mistral, or Llama model, and then adapting the model to your custom data. There are various techniques for fine-tuning, most of which involve modifying only a few parameters instead of modifying all of the parameters in the model. This is called *parameter-efficient fine-tuning (PEFT)*. For more information, see Hugging Face PEFT on GitHub.

The following are two common use cases when you might choose to fine-tune an LLM for a medical NLP task:

- **Generative task** – Decoder-based models perform generative AI tasks. AI/ML practitioners use ground truth data to fine-tune an existing LLM. For example, you might train the LLM by using MedQuAD, a public medical question-answering dataset. When you invoke a query to the fine-tuned LLM, you don't need a RAG approach to provide the additional context to the LLM.

- **Embeddings** – Encoder-based models generate embeddings by transforming text into numerical vectors. These encoder-based models are typically called *embedding models*. A *sentence-transformer model* is a specific type of embedding model that is optimized for sentences. The objective is to generate embeddings from input text. The embeddings are then used for semantic analysis or in retrieval tasks. To fine-tune the embedding model, you must have a corpus of medical knowledge, such as documents, that you can use as training data. This is accomplished with pairs of text based on similarity or sentiment to fine-tune a sentence transformer model. For more information, see Training and Finetuning Embedding Models with Sentence Transformers v3 on Hugging Face.

You can use Amazon SageMaker Ground Truth to build a high-quality, labeled training dataset. You can use the labeled dataset output from Ground Truth to train your own models. You can also use the output as a training dataset for an Amazon SageMaker AI model. For more information about named entity recognition, single label text classification, and multi-label text classification, see Text labeling with Ground Truth in the Amazon SageMaker AI documentation.

For more information about fine-tuning, see Fine-tuning large language models in healthcare in this guide.

Choosing an LLM

Amazon Bedrock is the recommended starting point to evaluate high-performing LLMs. For more information, see Supported foundation models in Amazon Bedrock. You can use model evaluation jobs in Amazon Bedrock in order to compare the outputs from multiple outputs and then choose the model that is best suited for your use case. For more information, see Choose the best performing model using Amazon Bedrock evaluations in the Amazon Bedrock documentation.

Some LLMs have limited training on medical domain data. If your use case requires fine-tuning an LLM or an LLM that Amazon Bedrock doesn't support, consider using Amazon SageMaker AI. In SageMaker AI, you can use a fine-tuned LLM or choose a custom LLM that has been trained on medical domain data.

The following table lists popular LLMs that have been trained on medical domain data.

LLM	Tasks	Knowledge	Architecture
BioBERT	Information retrieval, text classification,	Abstracts from PubMed, full-	Encoder

	and named entity recognition	text articles from PubMedCentral, and general domain knowledge	
ClinicalBERT	Information retrieval, text classification, and named entity recognition	Large, multi-center dataset along with over 3,000,000 patient records from electronic health record (EHR) systems	Encoder
ClinicalGPT	Summarization, question-answering, and text generation	Extensive and diverse medical datasets, including medical records, domain-specific knowledge, and multi-round dialogue consultations	Decoder
GatorTron-OG	Summarization, question-answering, text generation, and information retrieval	Clinical notes and biomedical literature	Encoder
Med-BERT	Information retrieval, text classification, and named entity recognition	Large dataset of medical texts, clinical notes, research papers, and healthcare-related documents	Encoder
Med-PaLM	Question-answering for medical purposes	Datasets of medical and biomedical text	Decoder

medAlpaca	Question-answering and medical dialogue tasks	A variety of medical texts, encompassing resources such as medical flashcards, wikis, and dialogue datasets	Decoder
BiomedBERT	Information retrieval, text classification, and named entity recognition	Exclusively abstracts from PubMed and full-text articles from PubMedCentral	Encoder
BioMedLM	Summarization, question-answering, and text generation	Biomedical literature from PubMed knowledge sources	Decoder

The following are best practices for using pretrained medical LLMs:

- Understand the training data and its relevance to your medical NLP task.
- Identify the LLM architecture and its purpose. Encoders are appropriate for embeddings and NLP tasks. Decoders are for generation tasks.
- Evaluate the infrastructure, performance, and cost requirements for hosting the pretrained medical LLM.
- If fine-tuning is required, ensure accurate ground truth or knowledge for the training data. Make sure that you mask or redact any personally identifiable information (PII) or protected health information (PHI).

Real-world medical NLP tasks might differ from pretrained LLMs in terms of knowledge or intended use cases. If a domain-specific LLM does not meet your evaluation benchmarks, you can fine-tune an LLM with your own dataset or you can train a new foundation model. Training a new foundation model is an ambitious, and often expensive, undertaking. For most use cases, we recommend fine-tuning an existing model.

When you use or fine-tune a pretrained medical LLM, it's important to address infrastructure, security, and guardrails.

Infrastructure

Compared to using Amazon Bedrock for on-demand or batch inference, hosting pretrained medical LLMs (commonly from Hugging Face) requires significant resources. To host pretrained medical LLMs, it's common to use an Amazon SageMaker AI image that runs on an Amazon Elastic Compute Cloud (Amazon EC2) instance with one or more GPUs, such as ml.g5 instances for accelerated computing or ml.inf2 instances for AWS Inferentia. This is because LLMs consume a large amount of memory and disk space.

Security and guardrails

Depending on your business compliance requirements, consider using Amazon Comprehend and Amazon Comprehend Medical to mask or redact personally identifiable information (PII) and protected health information (PHI) from training data. This helps prevent the LLM from using confidential data when it generates responses.

We recommend that you consider and evaluate bias, fairness, and hallucinations in your generative AI applications. Whether you are using a preexisting LLM or fine-tuning one, implement guardrails to prevent harmful responses. *Guardrails* are safeguards that you customize to your generative AI application requirements and responsible AI policies. For example, you can use Amazon Bedrock Guardrails.

Fine-tuning large language models in healthcare

The fine-tuning approach described in this section supports compliance with ethical and regulatory guidelines and promotes the responsible use of AI systems in healthcare. It is designed to generate insights that are accurate and private. Generative AI is revolutionizing healthcare delivery, but off-the-shelf models often fall short in clinical environments where accuracy is critical and compliance is non-negotiable. Fine-tuning foundation models with domain-specific data bridges this gap. It helps you create AI systems that speak the language of medicine while adhering to strict regulatory standards. However, the path to successful fine-tuning requires careful navigation of healthcare's unique challenges: protecting sensitive data, justifying AI investments with measurable outcomes, and maintaining clinical relevance in fast-evolving medical landscapes.

When lighter-weight approaches reach their limits, fine-tuning becomes a strategic investment. The expectation is that the gains in accuracy, latency, or operational efficiency will offset the significant compute and engineering costs required. It's important to remember that the pace of

progress in foundation models is rapid, so a fine-tuned model's advantage might last only until the next major model release.

This section anchors the discussion in the following two high-impact use cases from AWS healthcare customers:

- **Clinical decision support systems** – Enhance diagnostic accuracy through models that understand complex patient histories and evolving guidelines. Fine-tuning can help models deeply understand complex patient histories and integrate specialized guidelines This can potentially reduce model prediction errors. However, you need to weigh these gains against the cost of training on large, sensitive datasets and the infrastructure required for high-stakes clinical applications. Will the improved accuracy and context-awareness justify the investment, especially when new models are released frequently?
- **Medical document analysis** – Automate the processing of clinical notes, imaging reports, and insurance documents while maintaining Health Insurance Portability and Accountability Act (HIPAA) compliance. Here, fine-tuning may enable the model to handle unique formats, specialized abbreviations, and regulatory requirements more effectively. The payoff is often seen in reduced manual review time and improved compliance. Still, it's essential to assess whether these improvements are substantial enough to warrant the fine-tuning resources. Determine whether prompt engineering and workflow orchestration can meet your needs.

These real-world scenarios illustrate the fine-tuning journey, from initial experimentation to model deployment, while addressing healthcare's unique requirements at every stage.

Estimating costs and return on investment

The following are cost factors that you must consider when fine-tuning an LLM:

- **Model size** – Larger models cost more to fine-tune
- **Dataset size** – The compute costs and time increase with the size of the dataset for fine-tuning
- **Fine-tuning strategy** – Parameter-efficient methods can reduce costs compared to full parameter updates

When calculating the return on investment (ROI), consider the improvement in your chosen metrics (such as accuracy) multiplied by the volume of requests (how often the model will be used) and the expected duration before the model is surpassed by newer versions.

Also, consider the lifespan of your base LLM. New base models emerge every 6–12 months. If your rare disease detector takes 8 months to fine-tune and validate, you might only get 4 months of superior performance before newer models close the gap.

By calculating the costs, ROI, and potential lifespan for your use case, you can make a data-driven decision. For example, if fine-tuning your clinical decision support model leads to a measurable reduction in diagnostic errors across thousands of cases per year, the investment might quickly pay off. Conversely, if prompt engineering alone brings your document analysis workflow close to your target accuracy, it might be wise to hold off on fine-tuning until the next generation of models arrives.

Fine-tuning isn't one-size-fits-all. If you decide to fine-tune, the right approach depends on your use case, data, and resources.

Choosing a fine-tuning strategy

After you've determined that fine-tuning is the right approach for your healthcare use case, the next step is selecting the most appropriate fine-tuning strategy. There are several approaches available. Each has distinct advantages and trade-offs for healthcare applications. The choice between these methods depends on your specific objectives, available data, and resource constraints.

Training objectives

Domain-adaptive pre-training (DAPT) is an unsupervised method that involves pre-training the model on a large body of domain-specific, unlabeled text (such as millions of medical documents). This approach is well suited for improving the models' ability to understand medical specialty abbreviations and the terminology used by radiologists, neurologists, and other specialized providers. However, DAPT requires vast amounts of data and doesn't address specific task outputs.

Supervised fine-tuning (SFT) teaches the model to follow explicit instructions by using structured input-output examples. This approach excels for medical document analysis workflows, such as document summarization or clinical coding. *Instruction tuning* is a common form of SFT where the model is trained on examples that include explicit instructions paired with desired outputs. This enhances the model's ability to understand and follow diverse user prompts. This technique is particularly valuable in healthcare settings because it trains the model with specific clinical examples. The main drawback is that it requires carefully labeled examples. In addition, the fine-tuned model might struggle with edge cases where there aren't examples. For a instructions about

fine-tuning with Amazon SageMaker Jumpstart, see Instruction fine-tuning for FLAN T5 XL with Amazon SageMaker Jumpstart (AWS blog post).

Reinforcement learning from human feedback (RLHF) optimizes model behavior based on expert feedback and preferences. Use a reward model trained on human preferences and methods, such as proximal policy optimization (PPO) or direct preference optimization (DPO), to optimize the model while preventing destructive updates. RLHF is ideal for aligning outputs with clinical guidelines and making sure that recommendations stay within approved protocols. This approach requires significant clinician time for feedback and involves a complex training pipeline. However, RLHF is particularly valuable in healthcare because it helps medical experts shape how AI systems communicate and make recommendations. For example, clinicians can provide feedback to make sure that the model maintains an appropriate bedside manner, knows when to express uncertainty, and stays within clinical guidelines. Techniques such as PPO iteratively optimize model behavior based on expert feedback while constraining parameter updates to preserve core medical knowledge. This allows models to convey complex diagnoses in patient-friendly language while still flagging serious conditions for immediate medical attention. This is crucial for healthcare where both accuracy and communication style matter. For more information about RLHF, see Fine-tune large language models with reinforcement learning from human or AI feedback (AWS blog post).

Implementation methods

A *full parameter update* involves updating all model parameters during training. This approach works best for clinical decision support systems that require deep integration of patient histories, lab results, and evolving guidelines. The drawbacks include high compute cost and risk of overfitting if your dataset isn't large and diverse.

Parameter-efficient fine-tuning (PEFT) methods update only a subset of parameters to prevent overfitting or a catastrophic loss of language capabilities. Types include low-rank adaptation (LoRA), adapters, and prefix-tuning. PEFT methods offer lower computational cost, faster training, and are great for experiments such as adapting a clinical decision support model to a new hospital's protocols or terminology. The main limitation is potentially reduced performance compared to full parameter updates.

For more information about fine-tuning methods, see Advanced fine-tuning methods on Amazon SageMaker AI (AWS blog post).

Building a fine-tuning dataset

The quality and diversity of the fine-tuning dataset is critical to model performance, safety, and bias prevention. The following are three critical areas to consider when building this dataset:

- Volume based on fine-tuning approach
- Data annotation from a domain expert
- Diversity of data set

As shown in the following table, the dataset size requirements for fine-tuning vary based on the type of fine-tuning being performed.

Fine-tuning strategy	Dataset size
Domain adapted pre-training	100,000+ domain texts
Supervised fine-tuning	10,000+ labeled pairs
Reinforcement learning from human feedback	1,000+ expert preference pairs

You can use AWS Glue, Amazon EMR, and Amazon SageMaker Data Wrangler to automate the data extraction and transformation process to curate a dataset that you own. If you are unable to curate a large enough dataset, you can discover and download datasets directly into your AWS account through AWS Data Exchange. Consult your legal counsel prior to utilizing any third-party datasets.

Expert annotators with domain knowledge, such as medical doctors, biologists, and chemists, should be a part of the data curation process in order to incorporate the nuances of medical and biological data into the model output. Amazon SageMaker Ground Truth provides a low-code user interface for experts to annotate the dataset.

A dataset that represents the human population is essential for healthcare and life sciences fine-tuning use cases to prevent bias and reflect real world results. AWS Glue interactive sessions or Amazon SageMaker notebook instances offer a powerful way to iteratively explore datasets and fine-tune transformations by using Jupyter-compatible notebooks. Interactive sessions enable you to work with a choice of popular integrated development environments (IDEs) in your local

environment. Alternatively, you can work with AWS Glue or Amazon SageMaker Studio notebooks through the AWS Management Console.

Fine-tuning the model

AWS provides services such as Amazon SageMaker AI and Amazon Bedrock that are crucial for successful fine-tuning.

SageMaker AI is a fully managed machine learning service that helps developers and data scientists to build, train, and deploy ML models quickly. Three useful features of SageMaker AI for fine-tuning include:

- SageMaker Training – A fully managed ML feature that helps you efficiently train a wide range of models at scale
- SageMaker JumpStart – A capability that is built on top of SageMaker Training jobs to provide pre-trained models, built-in algorithms, and solution templates for ML tasks
- SageMaker HyperPod – A purpose-built infrastructure solution for distributed training of foundation models and LLMs

Amazon Bedrock is a fully managed service that provides access to high-performing foundation models through an API, with built-in security, privacy, and scalability features. The service provides the capability to fine-tune several available foundational models. For more information, see Supported models and Regions for fine-tuning and continued pre-training in the Amazon Bedrock documentation.

When approaching the fine-tuning process with either service, consider the base model, fine-tuning strategy, and infrastructure.

Base model choice

Closed-source models, such as Anthropic Claude, Meta Llama, and Amazon Nova, deliver strong out-of-the-box performance with managed compliance but limit fine-tuning flexibility to provider-supported options such as managed APIs like Amazon Bedrock. This constrains customizability, particularly for regulated healthcare use cases. In contrast, open-source models, such as Meta Llama, provide full control and flexibility across Amazon SageMaker AI services, making them ideal when you need to customize, audit, or deeply adapt a model to your specific data or workflow requirements.

Fine-tuning strategy

Simple instruction tuning can be handled by Amazon Bedrock model customization or Amazon SageMaker JumpStart. Complex PEFT approaches, such as LoRA or adapters, require SageMaker Training jobs or custom fine-tuning feature in Amazon Bedrock. Distributed training for very large models is supported by SageMaker HyperPod.

Infrastructure scale and control

Fully managed services, such as Amazon Bedrock, minimize infrastructure management and are ideal for organizations that prioritize ease of use and compliance. Semi-managed options, such as SageMaker JumpStart, offer some flexibility with less complexity. These options are suitable for rapid prototyping or when using pre-built workflows. Full control and customization come with SageMaker Training jobs and HyperPod, though these require more expertise and are best when you need to scale up for large datasets or require custom pipelines.

Monitoring fine-tuned models

In healthcare and life sciences, monitoring LLM fine-tuning requires tracking multiple key performance indicators. Accuracy provides a baseline measurement, but this must be balanced against precision and recall, particularly in applications where misclassifications carry significant consequences. The F1-score helps address class imbalance issues that can be common in medical datasets. For more information, see Evaluating LLMs for healthcare and life science applications in this guide.

Calibration metrics help you make sure that the model's confidence levels match real-world probabilities. Fairness metrics can help you detect potential biases across different patient demographics.

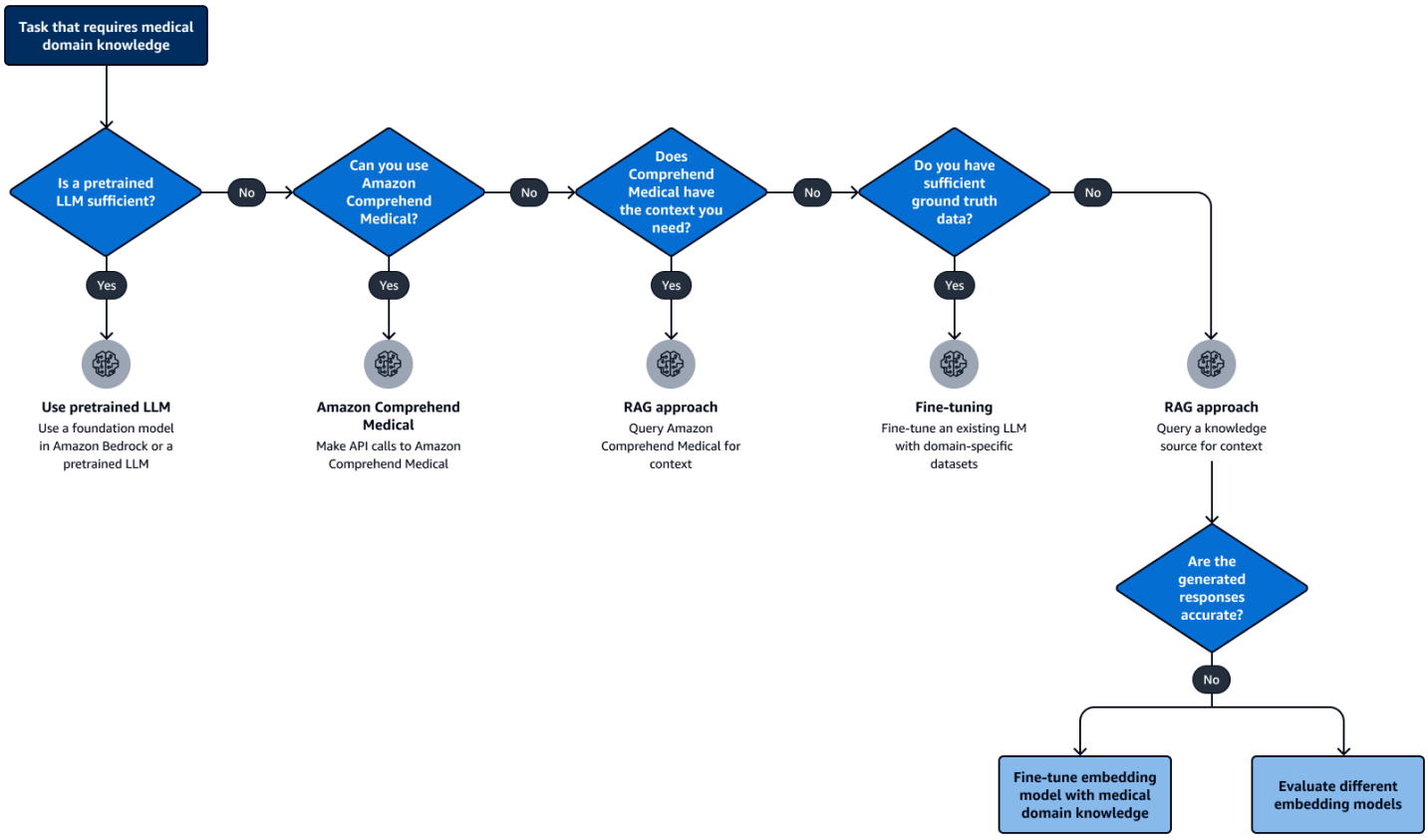
MLflow is an open source solution that can help you track fine-tuning experiments. MLflow is natively supported within Amazon SageMaker AI, which helps you to visually compare metrics from training runs. For fine-tuning jobs on Amazon Bedrock, metrics are streamed to Amazon CloudWatch so that you can visualize the metrics in the CloudWatch console.

Choosing an NLP approach for healthcare and life sciences

The Generative AI and NLP approaches for healthcare and life sciences section describes the following approaches for addressing natural language processing (NLP) tasks for healthcare and life science applications:

- Using Amazon Comprehend Medical
- Combining Amazon Comprehend Medical with an LLM in a Retrieval Augment Generation (RAG) workflow
- Using a fine-tuned LLM
- Using a RAG workflow

By evaluating the known limitations of LLMs for medical domain tasks and your use case, you can choose which approach will work best for your task. The following decision tree can help you choose an LLM approach for your medical NLP task:



The diagram shows the following workflow:

1. For healthcare and life science use cases, identify whether the NLP task requires specific domain knowledge. As needed, coordinate with subject matter experts (SMEs).
2. If you can use a general LLM or a model that has been trained on medical datasets, then use an available foundation model in Amazon Bedrock or the pretrained LLM. For more information, see Choosing an LLM in this guide.
3. If the entity detection and ontology linking capabilities of Amazon Comprehend Medical address your use case, then use the Amazon Comprehend Medical APIs. For more information, see Using Amazon Comprehend Medical in this guide.
4. Sometimes, Amazon Comprehend Medical has the required context but doesn't support your use case. For example, you might need different entity definitions, receive an overwhelming number of results, need custom entities, or need a custom NLP task. If this is the case, use a RAG approach to query Amazon Comprehend Medical for context. For more information, see Combining Amazon Comprehend Medical with large language models in this guide.
5. If you have a sufficient amount of ground truth data, fine-tune an existing LLM. For more information, see Customization approaches in this guide.
6. If the other approaches do not satisfy medical your NLP task objectives, implement a RAG solution. For more information, see Customization approaches in this guide.
7. After implementing the RAG solution, evaluate whether the generated responses are accurate. For more information, see Evaluating LLMs for healthcare and life science applications in this guide. It's common to start with an Amazon Titan Text Embeddings model or a general sentence transformer model, such as all-MiniLM-L6-v2. However, due to a lack of domain context, these models might not capture the medical terminology of the text. If necessary, consider the following adjustments:
   a. Evaluate other embedding models
   b. Fine-tune the embedding model with domain-specific datasets

Business maturity considerations

Business maturity is critical when adapting LLM solutions for healthcare and life science applications. These organizations face varying levels of complexity when implementing LLMs, depending on their acceptance criteria. Frequently, organizations that lack AI/ML resources invest

in contractor support to build LLM solutions. In these situations, it's important to understand the following trade-offs:

- **High performance for high cost and maintenance** – You might require a complex solution that involves fine-tuned or custom LLMs to meet stringent performance standards. However, this comes with higher costs and maintenance requirements. You might need to hire specialized resources or partner with contractors to maintain these sophisticated solutions. This can potentially slow development.
- **Good performance for low cost and maintenance** – Alternatively, you might find that services such as Amazon Bedrock or Amazon Comprehend Medical provide acceptable performance. Although these LLMs or approaches might provide perfect results, these solutions can often provide consistent, high-quality results. These solutions are lower cost and reduce the maintenance burden. This can accelerate development.

If a simpler, lower-cost approach consistently delivers high-quality results that meet your acceptance criteria, consider whether the increasing the performance is worth the cost, maintenance, and time tradeoffs. However, if the simpler solution falls significantly short of the target performance, and if your organization lacks the investment capacity for complex solutions and their maintenance requirements, consider postponing AI/ML development until more resources or alternative solutions are available.

In addition, for any medical NLP solution that relies on an LLM, we recommend that you perform continuous monitoring and evaluation. Assess feedback from users over time, and implement periodic assessments to make sure that the solution continues to meet your business objectives.

Evaluating LLMs for healthcare and life science applications

This section provides a comprehensive overview of the requirements and considerations for evaluating large language models (LLMs) in healthcare and life science use cases.

It is important to use ground truth data and SME feedback to mitigate bias and validate the accuracy of the LLM-generated response. This section describes best practices for collecting and curating training and test data. It also helps you implement guardrails and measure data bias and fairness. It also discusses the common medical natural language processing (NLP) tasks, such as text classification, named entity recognition, and text generation, and their associated evaluation metrics.

It also presents workflows for performing LLM evaluation during the training experimentation phase and post-production phase. Model monitoring and LLM operations are important elements of this evaluation process.

Training and test data for medical NLP tasks

Medical NLP tasks commonly use medical corpora (such as PubMed) or patient information (such as clinic patient visit notes) to classify, summarize, and generate insights. Medical personnel, such physicians, health care administrators, or technicians, vary in expertise and viewpoints. Due to subjectivity between these medical personnel, smaller training and test data sets pose a risk of bias. To mitigate this risk, we recommend the following best practices:

- When using a pretrained LLM solution, make sure that you have an adequate amount of test data. The test data should closely resemble the actual medical data. Depending on the task, this can range from 20 to more than 100 records.
- When fine-tuning an LLM, collect a sufficient number of labeled (ground truth) records from a variety of SMEs of the targeted medical domain. A general starting point is at least 100 high-quality records. However, given the complexity of the task and your accuracy acceptance criteria, more records might be required.
- If required for your medical use case, implement guardrails and measure data bias and fairness. For example, make sure that the LLM prevents misdiagnosis due to racial profiles of patients. For more information, see *Security and guardrails* in the Choosing an LLM section.

Many AI research and development companies, such as Anthropic, have already implemented guardrails in their foundation models to avoid toxicity. You can use toxicity detection to check input prompts and the output responses from LLMs. For more information, see Toxicity detection in the Amazon Comprehend documentation and see Guardrails in the Amazon Bedrock documentation.

In any generative AI task, there is a risk of hallucination. You can mitigate this risk by performing NLP tasks, such as classification. You can also use more advanced techniques, such as text similarity metrics. BertScore is a commonly adopted text similarity metric. For more information about techniques that you can use to mitigate hallucination, see A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models.

Metrics for medical NLP tasks

You can create quantifiable metrics after you establish ground truth data and SME-provided labels for training and testing. Checking quality through qualitative processes, such as stress testing and reviewing LLM results, is helpful for quick development. However, metrics act as quantitative benchmarks that support future LLM operations and act as performance benchmarks for each production release.

Understanding the medical task is critical. Metrics typically map to one of the following general NLP tasks:

- **Text classification** – The LLM categorizes the text into one or more predefined categories, based on the input prompt and provided context. An example is classifying a pain category by using a pain scale. Examples of text classification metrics include:
  - Accuracy
  - Precision, also known as *macro precision*
  - Recall, also known as *macro recall*
  - F1 score, also known as *macro F1 score*
  - Hamming loss
- **Named entity recognition (NER)** – Also known as *text extraction*, named entity recognition is the process of locating and classifying named entities that are mentioned in unstructured text into predefined categories. An example is extracting the names of medications from patient records. Examples of NER metrics include:
  - Accuracy

- Precision
- Recall
- F1 score
- Hamming loss

- **Generation** – The LLM is generates new text by processing the prompt and provided context. Generation includes summarization tasks or question-answering tasks. Examples of generation metrics include:
  - Recall-Oriented Understudy for Gisting Evaluation (ROUGE)
  - Metric for Evaluation of Translation with Explicit ORdering (METEOR)
  - Bilingual evaluation understudy (BLEU) (for translations)
  - String distance, also known as *cosine similarity*

FAQ about healthcare and life science use cases

The following are frequently asked questions related to using Amazon Comprehend Medical or LLMs for medical NLP tasks.

How do I choose between Amazon Comprehend Medical and an LLM?

If your task is to detect medical entities within your medical text, then review the Amazon Comprehend Medical documentation to understand which medical entities can be extracted and if any of the ontologies address your use case. If not, consider using an LLM. For more information, see Use cases for Amazon Comprehend Medical and Use cases for an LLM in this guide.

How can I provide Amazon Comprehend Medical results to an LLM?

You can incorporate Amazon Comprehend Medical results as context within your LLM prompts. This provides additional medical knowledge and terminology to the LLM. The provided context can improve the LLM's performance on tasks such as entity recognition, summarization, or question-answering. The guide provides several examples of how to structure prompts with Amazon Comprehend Medical results. For more information, see Combining Amazon Comprehend Medical with large language models in this guide.

What are some best practices when using Amazon Comprehend Medical with LLMs?

We recommend using the Amazon Comprehend Medical confidence scores to filter or prioritize entities within your prompts. It's also important to evaluate its performance on your specific data and validate that the entity definitions align with your requirements. Combining Amazon Comprehend Medical with domain-specific knowledge sources can further enhance the LLM's performance. For more information, see Best practices for using Amazon Comprehend Medical in a RAG workflow in this guide.

Should I use a pretrained medical LLM or fine-tune a general LLM for my healthcare use case?

The decision depends on your specific requirements and the availability of high-quality training data. Pretrained medical LLMs can provide a good starting point. However, you might still need to fine-tune them with your domain-specific data. If you have sufficient labeled data, fine-tuning a general LLM can be a viable option. For more information, see Choosing an LLM and Choosing an NLP approach in this guide.

How do I evaluate the performance of LLMs for medical NLP tasks?

We recommend using quantitative metrics, such as accuracy, precision, recall, and F1 score for text classification and named entity recognition tasks. You can use ROUGE and METEOR for text generation tasks. It's important to have reliable ground truth data labeled by subject matter experts and to implement processes for monitoring model performance over time. For more information, see Evaluating LLMs for healthcare and life science applications in this guide.

What are the trade-offs between high-complexity and low-complexity LLM solutions?

Fine-tuning an LLM or building a custom LLM are highly complex solutions. These approaches can improve performance but come with higher costs and maintenance requirements. Simpler solutions, such as using pretrained LLMs or Amazon Comprehend Medical, might provide acceptable performance with lower costs and faster development cycles. However, these approaches might not meet stringent accuracy requirements for some use cases. For more information, see Business maturity considerations in this guide.

Next steps and resources

This guide helps you use AWS services to automate medical NLP and generative AI tasks for real-world applications in production environments. It describes how you can use Amazon Comprehend Medical, supported LLMs in Amazon Bedrock, pretrained medical LLMs, or fine-tuned LLMs to achieve your healthcare and life science business objectives. This guide describes the advantages and limitations for the following approaches:

- Using Amazon Comprehend Medical independently
- Providing Amazon Comprehend Medical results to an LLM
- Using a pretrained general LLM or a medical LLM in a Retrieval Augmented Generation (RAG) approach
- Fine-tuning a general LLM or medical LLM

Use the decision tree and the business maturity considerations in this guide to choose between these approaches based on your organization's AI/ML maturity level. Although Amazon Comprehend Medical and Amazon Bedrock LLMs provide powerful capabilities, they are only successful if you properly implement and evaluate them. Use the evaluation information and metrics described in this guide to validate the performance of your solution.

For next steps, we recommend that healthcare IT managers, architects, and technical leads work with AI/ML practitioners to identify their NLP medical task. Use this guide to choose a development path, and then use the appropriate AWS services and features to successfully implement an automated solution on AWS.

AWS resources

- Amazon Comprehend Medical documentation:
  - Developer Guide
  - API Reference
- Amazon Bedrock documentation
  - Amazon Bedrock model evaluation
  - Fine-tuning in Amazon Bedrock
- Fine-tune a model in Amazon SageMaker AI

- Amazon SageMaker Ground Truth
- Amazon Comprehend toxicity detection
- AWS Healthcare Competency Partners

Other resources

- Open Medical-LLM Leaderboard
- A Survey of Large Language Models for Healthcare: from Data, Technology, and Applications to Accountability and Ethics
- Large Language Models Are Poor Medical Coders — Benchmarking of Medical Code Querying
- From Beginner to Expert: Modeling Medical Knowledge into General LLMs

Contributors

Authoring

- Joe King, AWS Senior Data Scientist
- Ankith Ede, AWS Solutions Architect
- Clement Perrot, AWS Senior Generative AI Strategist
- Jillian Forde, AWS Senior Solutions Architect
- Rajesh Sitaraman, AWS Senior Delivery Consultant
- Ross Claytor, AWS Principal Applied Scientist
- Shivesh Ummat, AWS Solutions Architect

Reviewing

- Dilshad Raihan Akkam Veettil, AWS Senior Data Scientist
- Joseph Cottingham, AWS Deep Learning Architect

Technical writing

- Lilly AbouHarb, AWS Senior Technical Writer

Document history

The following table describes significant changes to this guide.

Change	Description	Date
New sections	We added the Fine-tuning large language models in healthcare section and the Prompt engineering section.	December 5, 2025
Initial publication	—	December 16, 2024