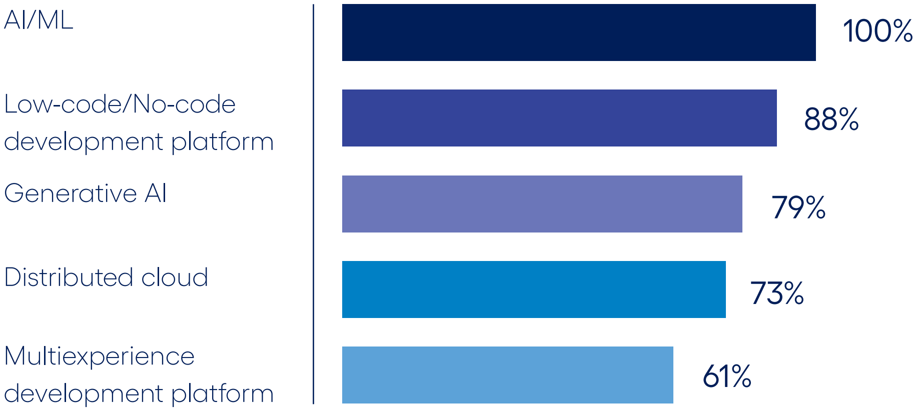
Revolutionizing claims adjudication using generative AI

Contents

Executive summary

The US healthcare industry faces increasing inflation and labor costs, prompting CIOs, IT leaders and business professionals to seek cost reductions and efficiency improvements. This environment has accelerated innovation, with generative AI (gen AI) emerging as a significant disruptor. According to the 2024 Gartner report, AI/ML is expected to be a top priority by 2026, with 79% of respondents identifying gen AI as a major game changer in the next three years.

Technologies most likely to be implemented by 2026



Top game-changing technologies in the next three years:

Generative AI

Source: https://emt.gartnerweb.com/ngw/globalassets/en/information-technology/images/infographics/2024-cio-agenda/2024-cio-agenda-infographics/2024-cio-agenda-banking-and-investment-infographic

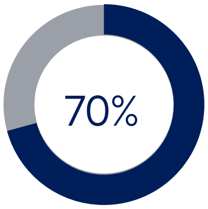
However, the adoption of gen AI introduces risks, including heightened cybersecurity threats and ethical challenges. Despite these concerns, our comprehensive strategy balances innovation with robust safeguards. This whitepaper presents a detailed solution for implementing gen AI in claims adjudication, addressing the associated risks and mitigation strategies.

Introduction

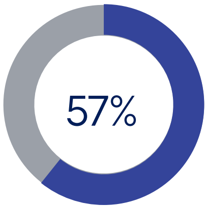
Global CEOs are making gen AI a top investment priority. In 2023, a survey was conducted by KPMG, encompassing 2,100 C-suite leaders from 16 companies across nine industries that include energy, education, financial services, government, healthcare, industrial manufacturing, life sciences, tech, retail and consumer packaged goods. Each of these companies has more than US$500 million in annual revenue. The survey results show that 70% of companies are investing heavily in gen AI—to ensure having a competitive edge in the future. At least 52% expect to see a return on investment (ROI) in three to five years. In fact, increased profitability was cited by 22% respondents as the number one benefit of implementing gen AI within an organization.

While global CEOs are willing to go ahead and make investments in gen AI, they also recognize the need to address risks that come with gen AI. 82% of respondents believe that gen AI has heightened cybersecurity risks by providing new attack strategies for adversaries. Additionally, 57% cited ethical challenges as the top concern, followed closely by a lack of regulation and integration complexity.

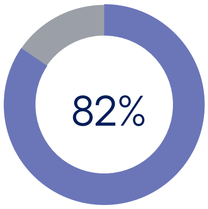
Perspectives from CEOs on generative AI



Generative AI is their top investment priority



Ethical challenges—the number one concern when it comes to implementing generative AI



AI may provide new attack strategies for adversaries

Source: https://kpmg.com/dk/en/home/insights/2023/10/kpmg-ceo-outlook-2023.html

As we address the concerns raised by global CEOs, it is important to understand what gen AI is and how we can reduce upfront investments in gen AI.

Gen AI is a type of artificial intelligence technology that uses pretrained large language models (LLMs) that can produce several types of content, including text, imagery, audio and code or synthetic data. LLMs use the transformer architecture, a type of neural network, which has two components—an encoder and a decoder. The encoder takes the input data, extracts meaning and context from the data and passes it on to the decoder, which then generates the output.

To reduce the upfront investment in gen AI, Cognizant has partnered with Amazon, Microsoft and Google to leverage their cloud computing platforms, which have inbuilt foundational models. This eliminates the need to build an LLM from scratch, thereby exponentially reducing the upfront hardware investments and build time. These platforms provide additional features to develop custom models. Their usage is on a pay-as-you-go basis, which shifts costs from a CapEx to an OpEx model. In addition, leveraging open-source AI frameworks and libraries such as LangChain and Hugging Face Transformers can significantly reduce development and licensing costs.

Another approach that Cognizant has adopted is the identification of specific areas where proof of concepts (POC) can be implemented. The success in these limited deployments will test gen AI's effectiveness and hence justify the need for future investments. To become more cost-effective, instead of hiring external talent, Cognizant has invested significantly in upskilling its existing workforce with primary focus on the in-house data science and automation teams.

To mitigate data privacy and security risks associated with gen AI, Cognizant has adopted a comprehensive strategy that balances innovation with robust safeguards. The measures we take include the following:

- Enabling data encryption at rest and transit, and role-based access control (RBAC) features, which are a part of the cloud platforms.
- Integrating privacy and data protection principles into the design and development of custom LLMs by anonymizing or pseudonymizing PHI data.
- Conducting regular data privacy impact assessments (DPIAs)—to identify and mitigate potential risks.
- Establishing a cross-functional AI ethics committee responsible for overseeing gen AI projects, ensuring they align with the ethical standards, and managing data privacy concerns.
- Developing and maintaining a comprehensive incident response plan for gen AI-related data breaches. This plan includes clear steps for containment, investigation, communication and remediation—and will be regularly tested for effectiveness and readiness through tabletop exercises and simulations.

The challenge:

Rising cost and complexity of claims management

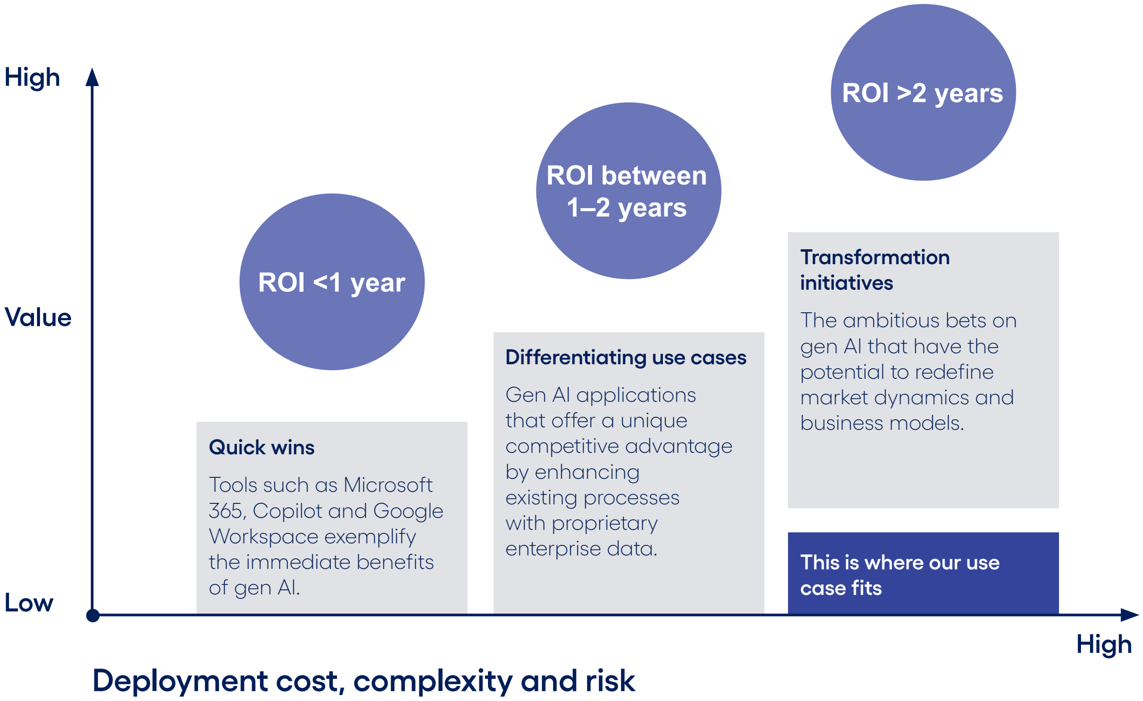
According to the American Medical Association (AMA), the US healthcare industry loses approximately US$210 billion annually due to inefficient claims processing. A significant portion of these costs stem from the manual review of complex claims, accounting for 15%–20% of all claims.

Automating these claims using gen AI has the potential to reduce costs by 30%–70%. However, building a custom gen AI model requires substantial upfront investments in infrastructure, token costs, training time and expert resources. To justify these costs, it's essential to measure the potential ROI. Gartner identifies three primary categories of gen AI investments, each with unique objectives and evaluation criteria.

Our approach

Our opportunity falls under the category of transformational initiatives (see the diagram below), and based on this, we have estimated ROI within 9 months to 12 months from go-live for a single client, and another 12 months to 24 months across all clients for whom we process claims. In total, we're looking at a timeframe of two-and-a-half years to three years to deploy gen AI across the entire claims business.

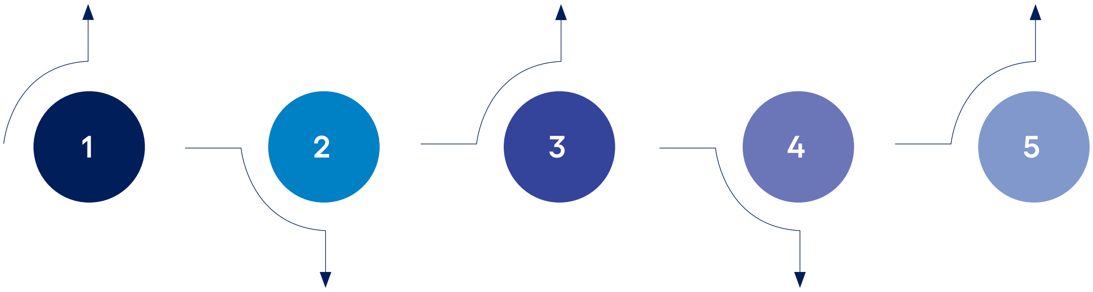
Generative AI use case categories


High
Value
Low
ROI <1 year
ROI between 1–2 years
ROI >2 years
Quick wins
Tools such as Microsoft 365, Copilot and Google Workspace exemplify the immediate benefits of gen AI.
Differentiating use cases
Gen AI applications that offer a unique competitive advantage by enhancing existing processes with proprietary enterprise data.
Transformation initiatives
The ambitious bets on gen AI that have the potential to redefine market dynamics and business models.
This is where our use case fits
High
Deployment cost, complexity and risk

Source: https://www.gartner.com/en/articles/take-this-view-to-assess-roi-for-generative-ai

We leverage a large language foundational model and customize it with domain-specific claims data and SOPs—to autoprocess the edit codes in the claims that are currently manually adjudicated. We follow a structured model as depicted in the diagram below.

Gen AI solution for claims adjudication



Service and model selection

Determine which platform— such as Open AI, Azure Open AI and Amazon Bedrock along with the foundation LLM such as ChatGPT—to be used for building the model.

Data collection, preparation and ingestion

Gather and generate claims data and SOPs, and preprocess it to efficiently organize and structure data, ensuring optimal performance in location answers within our application.

Customized model creation

Create the POC model, customized on domain data, and improve its performance using prompt engineering and retrieval augmented generation.

POC model evaluation

If the model meets 70%–80% of the expected performance, then fine-tune the model.

Fine-tuning the model

Fine-tune the POC model and deploy it in production as and when it meets the expected performance. Thereafter, further refine the model through continuous feedback.

Our detailed solution

Following is a detailed description of our gen AI solution for claims adjudication.

Service and model selection

We first evaluated the most suitable platform among OpenAI, Azure OpenAI and Amazon Bedrock. See the table below for an overview of the attribute-wise comparison of these three services.

Service selection

Features	OpenAI	Amazon Bedrock	Azure OpenAI
Integration ease	High	Medium	High
Scalability	High	Very high	High
Pretrained models	Extensive	Limited	Extensive
Custom model training	Yes	Yes	Yes
Data security	Low	High	High
Regulatory compliance	Low	Medium	High
Prompts and completions	Stores prompts and completions to train, retrain or improve the models. Therefore, your data might be used to train other OpenAI models.	Doesn't use your prompts and continuations to train any AWS models or distribute them to third parties. Amazon Bedrock has the concept of a model deployment account in each AWS region, where Amazon Bedrock is available. There is one such deployment account per model provider. These accounts are owned and operated by the Amazon Bedrock service team. Model providers don't have any access to those accounts.	Stores prompts and completions data for a maximum of 30 days in the same region, which is encrypted and only available to authorized employees. Therefore, your data cannot be used to train other OpenAI models.
Pricing	Pay-as-you-go	Pay-as-you-go	Pay-as-you-go

From this evaluation, we concluded that though OpenAI is the pioneer in developing foundational LLMs, it is not a good proposition with the advent of Amazon Bedrock and Microsoft Azure OpenAI, for enterprise usage. This is primarily because of the lack of features that provide data security and regulatory compliance compared to its counterparts. On the contrary, with Amazon Bedrock and Azure OpenAI, you will have full control over the data you use to customize the model for your solution. However, Azure OpenAI's performance on response time and regulatory compliance is better compared to Amazon Bedrock, and these are two very critical parameters for the problem that we are trying to solve. Hence, we selected Azure OpenAI to build our model.

Azure OpenAI service provides access to OpenAI's powerful language models such as GPT-4 and GPT-3.5-Turbo. Users can access the service through a web-based interface in the Azure OpenAI studio. We evaluated ChatGPT-3.5 Turbo 16K and ChatGPT-4 32K on attributes that are most critical to this opportunity. The table given below presents the details of the comparison.

Model selection

Attributes	GPT-3.5 Turbo16K	GPT-4 32K	Recommended
Speed of response	Speed of response	Slow at giving answers	GPT-3.5 Turbo16K
Cost	Less expensive	More expensive	GPT-3.5 Turbo16K
Context length	16,384 tokens	32,768 tokens	GPT-3.5 Turbo16K, because GPT-4 32K will have a limit of 24,000 words or 48 pages which is not needed
Memory and computational power	Operates effectively with less computational power and memory	Requires more computational power and memory	GPT-3.5 Turbo16K
Visual input	Does not have this capability	Has this capability	Not relevant because the claims data is available in the EDI gateway

The above evaluation helped us zero in on ChatGPT-3.5 Turbo 16K as the foundational LLM for us to use and build the solution. Before we can generate text or inference, we deploy this foundational model in Azure OpenAI Studio. Once this deployment is completed, we proceed to build our customized model.

Data collection, preparation and ingestion

The core of a successful LLM lies in a well-curated and diverse dataset. A high-quality dataset is essential for producing coherent and contextually relevant output. Some of the key aspects of these datasets are type and size of data, data sources and data quality.

For our model, we use the following data:

- **Claims:** Claims: Historical (processed) and new-day claims data for only those claims with the five shortlisted edit codes for the POC. We use an estimated number of 10,000 claims and this data is obtained from the FACETS team.
- **SOPs:** We get the five shortlisted edit codes from the client knowledge management system/repository.
- **Data tables:** The FACETS team provides this information. It is needed to establish the relationship between those tables (provider, member, pricing, etc.) and process the claims with the shortlisted edit codes.

Once we have the data, it is important to preprocess it to create a pretraining corpus for the LLM to remove noise, redundancy and irrelevance. This is because the data quality can significantly impact the capacity and performance of the model. Data preprocessing has the following four major steps:

Data cleaning and ingestion:

It involves identifying and rectifying inaccuracies, inconsistencies and irrelevant elements within raw data. Common cleaning procedures for claims data include removing duplicate entries, handling missing or erroneous values and addressing formatting irregularities. Azure OpenAI does not support .xlsx and .csv formats, so we convert the claims data, which can be in any one of these formats, into .json format. Text-specific cleaning tasks, applicable for SOPs, involve removing special characters, punctuation, stop words and screenshots. In addition, instead of having five SOPs, we can have one business requirements document (BRD) for the five edit codes. This will result in a faster build time with reduced cost for the gen AI engine.

After the data is preprocessed and made ready for usage, we ingest the data into Azure Blob Storage, which is a highly scalable and reliable cloud storage solution and is ideal for storing large amounts of unstructured data. We then connect it to Azure OpenAI for ingestion via HTTP or HTTPS. We add our data in Azure Blob storage account through the Azure OpenAI Studio, in the Chat Playground.

Tokenization:

LLMs process text data in units called tokens, which can be words, parts of words, or even characters. Tokenization is the process of breaking down text into these tokens. This method is particularly suitable for SOPs, which often contain many specialized terms. There are three tokenization methods—word level, character level and subword level. We use subword-level tokenization for the POC that balances the need to represent both common and rare words effectively. This method is particularly suitable for SOPs, which often contain many specialized terms.

Chunking:

It involves breaking down the data into digestible portions and sending only the most relevant chunks to the model. This way, we get the precise insights we want. Chunk size determines what embedding models should be used. For our POC, we use large chunk size and top-k retrieved chunks along with a range of chunk sizes (we start by exploring with 512 or 1024 tokens), to limit how much data we can input into our LLM. Out of the various chunking methods, we use variable size chunking because we need intact text or passages and larger chunks. The variable chunking does this by preserving the sentence structure, thereby producing better results. It will partition the SOPs based on content characteristics such as end-of-sentence punctuation marks, end-of-line markers, headers, etc.

To tokenize and chunk input data in Azure OpenAI, we use LangChain, which simplifies the process of loading documents, estimating token counts and splitting text into manageable chunks. For PDF SOPs, we use TikToken to estimate token counts. TikToken uses byte pair encoding, a technique that efficiently represents common words as single tokens and breaks down rare words into smaller subword tokens. This approach aligns with subword-level tokenization and ensures optimal processing by the LLM.

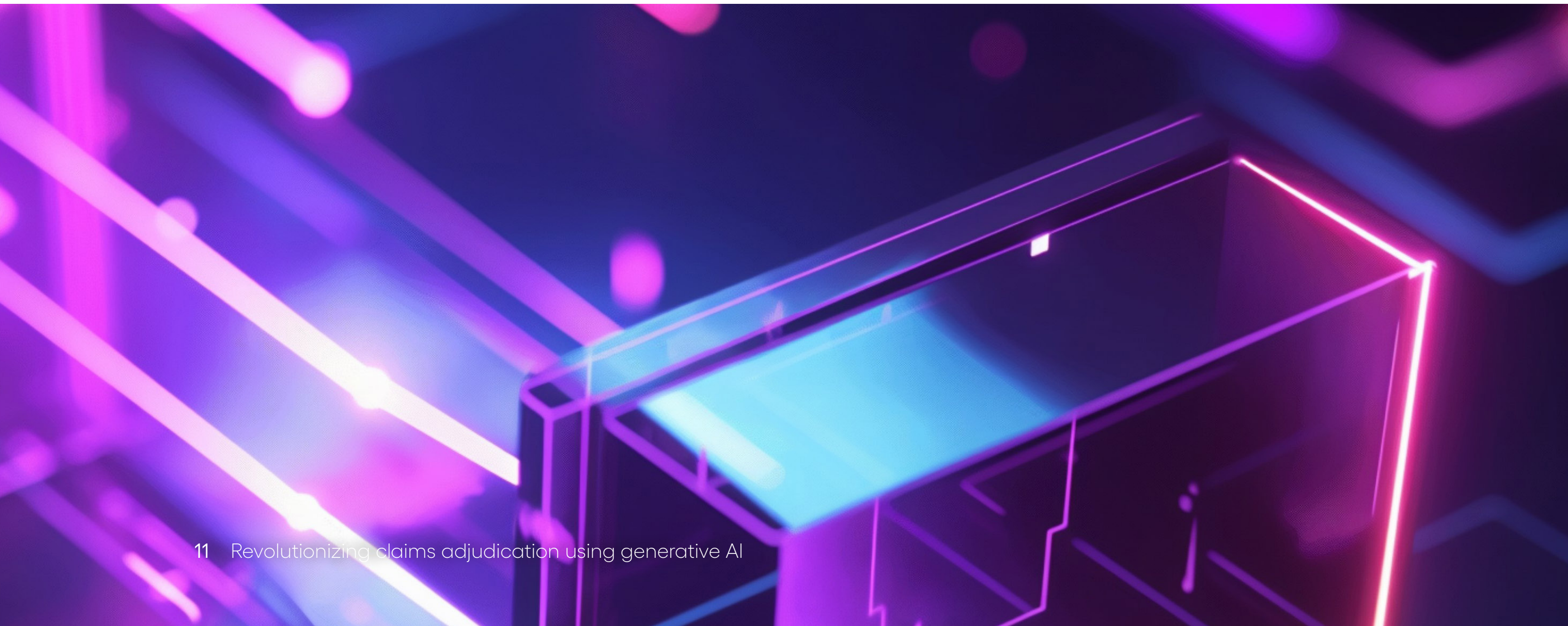
Embeddings:

Once the text is tokenized, it needs to be converted into numerical representations called embeddings. These embeddings allow the model to understand the semantic meaning of words and their relationships. Azure OpenAI's text-embedding-ada-002 model is used to generate these embeddings for our project.

Customized model creation:

Once the foundational ChatGPT-3.5 Turbo 16K model is deployed in Azure OpenAI, it must be customized as it does not contain the healthcare specific data needed to adjudicate claims. Customization can be done using:

- Prompt engineering
- Retrieval augmented generation (RAG)



Prompt engineering:

It is the art of crafting instructions to guide LLMs toward specific responses. We'll use chain-of-thought prompting to guide ChatGPT-3.5 Turbo 16K toward the desired output. Since we use custom data, few-shot prompting won't be applicable.

To minimize the number of iterations to get the desired or nearly desired output, we use the following techniques:

- System message: It starts with a system message at the beginning of the prompt. It primes the model by offering context, instructions or relevant information. For instance, a system message could define the assistant's personality and specify response formats.
- CoT prompting: This technique involves breaking down a task into smaller and step-by-step instructions. This approach can improve the accuracy and clarity of the model's responses. We implement this by adding a specific instruction to the user message field on Azure OpenAI Studio Chat Playground.
- Structured output: It specifies the desired output format in the prompt, in the user message field on Azure OpenAI Studio Chat Playground. It guides the model to produce responses that match your requirements, making parsing easier.
- Temperature, top_p and other parameters: The parameters such as temperature and top_p control the randomness of the model's output. Temperature determines how randomly the model predicts words, while top_p controls the length of the word list. This is why it is not advisable to play or change the values of both parameters simultaneously. For our solution, we adjust the temperature to 0.1, to prioritize precise responses. Other parameters such as strictness and retrieved documents can also be fine-tuned—to control the quality and relevance of the output. By understanding and effectively using these parameters, we can optimize the model's performance and cost-efficiency.

The above system message and user message are inputs for the corresponding fields in the Azure OpenAI Studio Chat Playground. However, this will be the backend entity and the claims processor will have to input the 118-token prompt below on the UI that will be created to ensure seamless interaction between the claim processor and the LLM.

"You are a claim processor. Your task is to process claims by resolving the edits and warning messages on a claim and provide recommendation on whether to pay, deny or pend the claim. Analyze the edit/s on this claim by referring to the relevant SOPs, validate the rules and provide your recommendation. Explain the step-by-step approach for your recommendation. In your recommendation, provide the reason to pay, deny or pend the claim." If you are unable to process an edit or edits on a claim mention that you don't know the answer , this is the standard prompt that applies to all edit codes in claims and serves as the input to the LLM to generate the output.

To create a user-friendly solution, a React.js-based application is developed. The UI includes three main tabs:

- **Claim information:** Displays detailed claim information, including provider, member, billing elements, authorization details and any edits or warnings.
- **User query:** Allows users to input prompts or questions related to the claim.
- **Output:** Presents the LLM-generated response or recommendation. A select group of SMEs will have access to a review pane—to validate the accuracy of the LLM's output, contributing to the model's improvement through RLHF. The findings of this review are presented in a tabular format and look something like this:

Claim ID	**Edit code/s warning message/s**	**Reviewed by**	**Accurately processed (Y/N)**	**Remarks if N**

The review process involves auto populating claim ID, edit codes and reviewer information. SMEs mark a claim as accurately processed or not. If accurate, the remarks section is disabled; otherwise, SMEs must provide details about the error.

The model performance pane provides insights into the model accuracy and the list of edit code-wise steps where the LLM has failed. With prompt engineering for the POC, we look at a minimum model accuracy in the range of 50%–60%, with a prompt iteration limit of five to seven.

RAG

It is an LLM learning technique that merges the retrieval mechanisms and generative capabilities to enhance the performance of large language models. It enables LLMs to give relevant and domain-specific responses versus generalized responses. In RAG architecture, there is no extra training. The LLM is pretrained using public data, but it generates responses that are augmented by information from the retriever. So, it uses the claims data and provides it as part of the prompt. Then it is used to query the LLM model and retrieve relevant data, and applied as an augmented context for the LLM. It is highly reliable when dealing with sensitive data such as PHI in healthcare. RAG can assist claims processors in decision making by providing up-to-date information and processing guidelines. The below diagram provides a view on the three elements of RAG working.

Understanding the basics of how RAG works

1. Retriever

Responsible for the initial step of retrieving relevant information from data sources. It uses retrieval techniques such as keyword-based search, document retrieval or structured database queries to fetch pertinent data.

Primary goal is to compile a set of contextually relevant information that can be used to enrich the user's query.

2. Ranker

Refines the retrieved information by assessing its relevance or importance. It assigns scores or ranks to the retrieved data points, helping prioritize the most relevant ones.

Ensures the most pertinent information is presented to the generator for content generation.

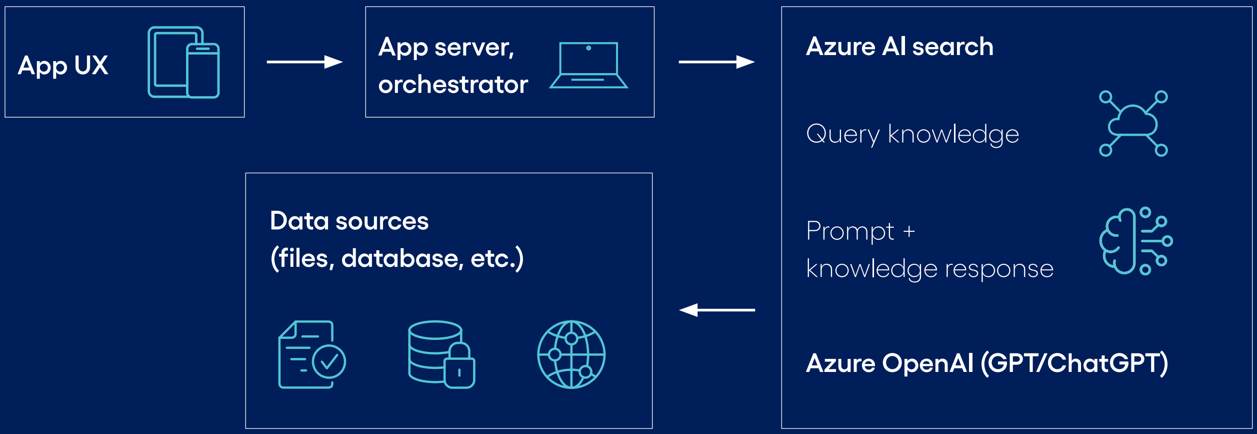
3. Generator

Takes the retrieved and ranked information, along with the user's original query and generates the final response and output.

Ensures that the response aligns with the user's query and incorporates the factual knowledge retrieved from various data sources.

Unlike traditional databases for RAG implementation, we use a vector database to store vectors (fixed-length lists of numbers) along with other data items. This is primarily because vector databases outperform traditional databases as they do not require a predefined schema, can handle structured and unstructured data, and provide similarity search. For our use case, we use Azure AI Search as the vector database. The below diagram is an illustration of how the four components in the RAG pattern—app UX (which provides the user experience), app server or orchestrator (which is the integration code that coordinates the handoffs between information retrieval and the LLM), Azure AI Search (which is the information retrieval system) and the LLM (which in this case is ChatGPT 3.5 Turbo 16K)—work together to generate a response.

RAG architecture in Azure AI search

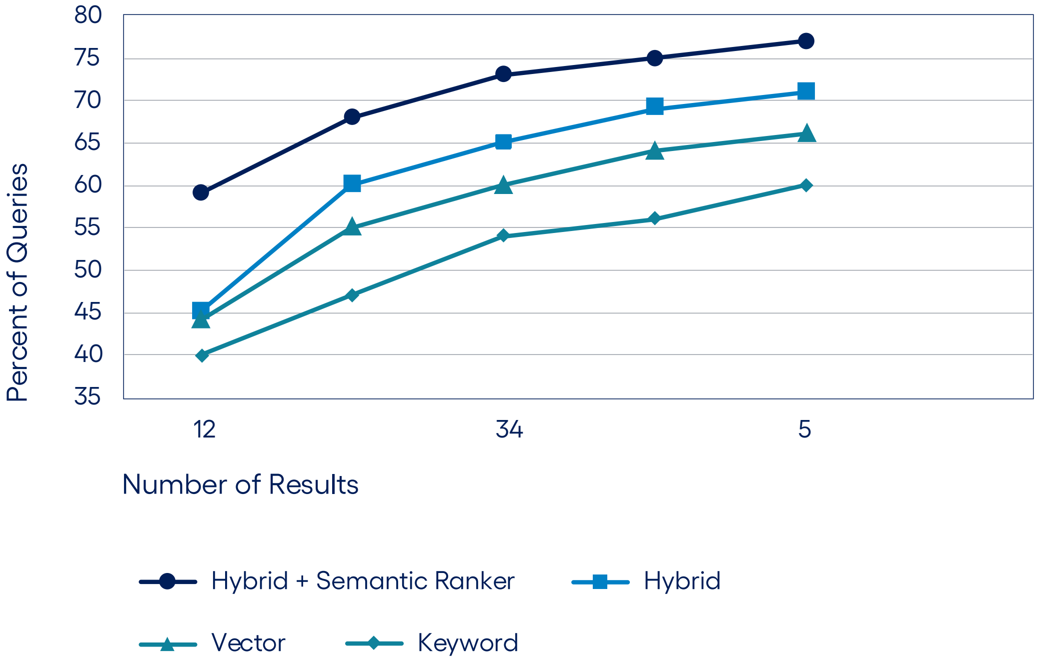

App UX
App server, orchestrator
Azure AI search
Query knowledge
Prompt + knowledge response
Azure OpenAI (GPT/ChatGPT)
Data sources (files, database, etc.)

In Azure AI Search, all searchable content is stored in a search index that's designed for fast queries with millisecond response time as its internal data structures exist to support that objective. When we set up the data for our RAG solution, we use the features that create and load an index in Azure AI Search. An index includes fields that duplicate or represent our source content. An example of an index field in our case will be the edit codes. Once our data is in a search index, we use the query capabilities of Azure AI Search to retrieve content. The query stack in Azure AI Search has two main layers of execution—retrieval and ranking.

- **Retrieval:** Often called L1, the goal of this step is to quickly find all the documents from the index that satisfy the search criteria—possibly across millions or billions of documents. These are scored to pick the top few (typically in the order of 50) to return to the user or to feed to the next layer. Azure AI Search supports three different L1 modes—keyword, vector and hybrid.
  - **Keyword:** Uses traditional full-text search methods, where the content is broken into terms through language-specific text analysis. The inverted indexes are created for fast retrieval and the BM25 probabilistic model is used for scoring.
  - **Vector:** Enables you to find documents that are similar to a given query input which is based on the vector embeddings of the content.
  - **Hybrid:** Performs both keyword and vector search and applies a fusion step to select the best results from each technique. Azure AI Search currently uses reciprocal rank fusion (RRF) to produce a single result set. Hybrid search brings out the best of keyword and vector search.
- **Ranking:** Also called L2, it takes a subset of the top L1 results and computes higher quality relevance scores to reorder the result set. It is critical for RAG applications to make sure the best results are in the top positions. L2 can improve L1's ranking because it applies more computational power to each result using semantic ranking. In Azure AI Search, semantic ranker is a feature that measurably improves search relevance by using Microsoft's language understanding models to rerank search results. It is a collection of query-side capabilities that improve the quality of the initial BM25-ranked search result for the input query. This feature provides the user with the top search results by reranking the existing result set, consisting of the top 50 results as scored by the BM25 ranking algorithm.

The semantic ranker can rank the top 50 results from L1. While hybrid search yields the best results, when it is combined with semantic ranking, it further enriches the search results. This is because semantic ranking puts the top three to five results at the top. Thus hybrid + semantic ranking finds the best content for the LLM at each result set size as indicated in the graph below:

Hybrid retrieval with semantic ranking outperforms vector-only search



In our case, we use hybrid search and semantic ranker. With RAG implementation for the POC, we aim for a minimum model accuracy in the range of 70%-80%.

POC model evaluation

To evaluate the performance of an LLM, we use accuracy as the metric. This will be done through a manual evaluation approach or RLHF, where the SME manually reviews the model's generated outputs. The evaluation could yield the following two possibilities:

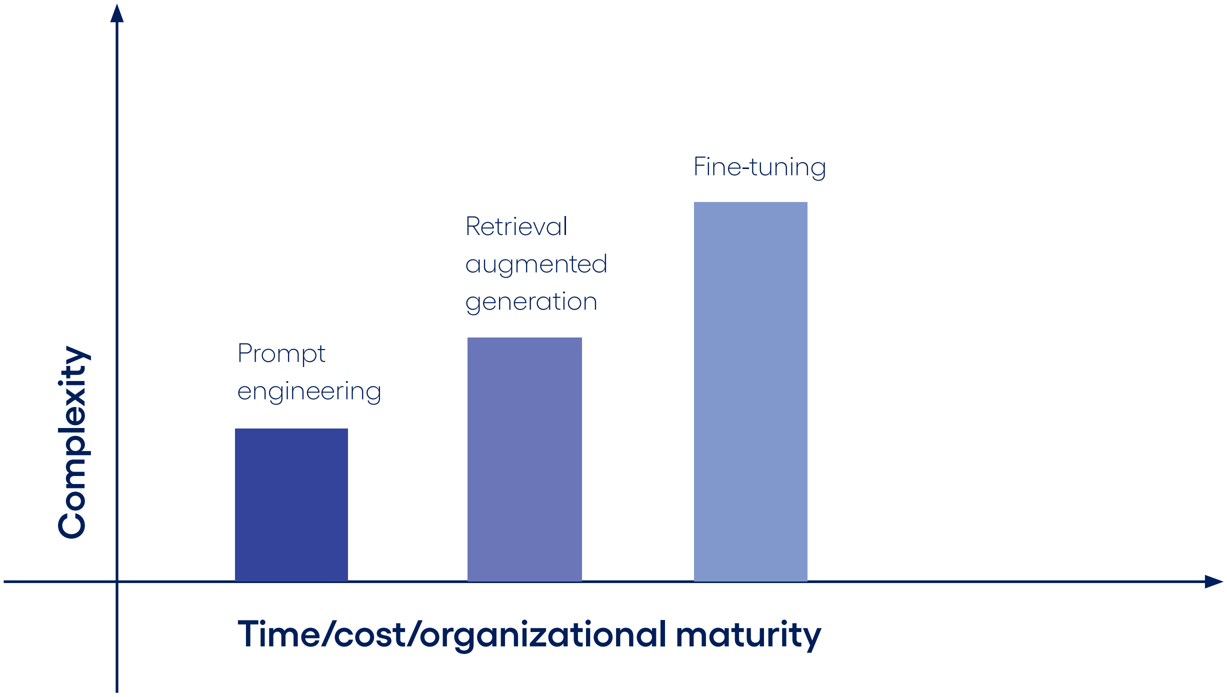
- Using prompt engineering, if we achieve 95% model accuracy, then we deploy it in production. Otherwise, we target at least 50%–60% with five to seven iterations of the prompt.

- Then we implement RAG and if we are able achieve 95% model accuracy, we deploy the model. However, if we don't achieve the expected accuracy, then we will target at least 70%-80% and proceed to fine-tuning the model.

Fine-tuning the model

Fine-tuning is a process to improve a pretrained model's performance on specific tasks such as processing claims. It involves training the model on a smaller and specific data set. While it can significantly enhance the model's accuracy, it's a costly and time-consuming process. The cost of training a ChatGPT-3.5 Turbo 16K model depends on various factors, including the size of the training dataset and the number of training epochs. A critical point to go ahead with fine-tuning the model is to have a baseline performance which we have from the POC model. This will help determine if fine-tuning has improved the model's performance. This is specifically the reason why we follow the below road map to build a customized model, taking into consideration the complexity and time/cost/organizational maturity.

Fine-tuning the model



In Azure OpenAI, we have the following two steps for fine-tuning.

- **Upload your dataset:** We recommend at least hundred excellent quality samples to start with our use case and then keep on increasing the size of the dataset till we achieve the expected model accuracy.

- **Create custom model:** Azure OpenAI service employs the supervised fine-tuning method called LoRA, or low rank approximation, to fine-tune models in a way that reduces their complexity without significantly affecting their performance. This method allows us to fine-tune the model efficiently by adjusting only a small subset of its parameters, rather than the entire model. This significantly reduces training time and cost. We utilize Azure OpenAI Studio to implement LoRA and fine-tune our model. Once the fine-tuning process is complete, we deploy the customized model and evaluate its performance using RLHF. If the model doesn't achieve the desired accuracy of 95%, we iteratively add more training data to refine its capabilities and help reach the target performance.

Now, to deploy this model, we have two options to choose from. The first one is to have the model integrated with the claims adjudication system. The second one is to deploy it outside this system but within the client environment.

The first approach will result in an increase in auto adjudication percentage. However, the health plans will have to factor in additional budgets and capital expenses to modernize their existing adjudication platforms. Also, it presents several technical and operational challenges, which generally revolve around security, performance, integration complexity and compliance.

So, we adopt the second approach, where the gen AI model is hosted outside the adjudication engine but within the client's environment. For the second approach, we deploy an RPA bot outside the claims adjudication platform—to segregate the claims with the shortlisted five edit codes for the POC.

One of the key challenges that we encounter once the model is deployed to production is SOP updates. For this, we first determine the frequency as well as the impact of the updates and then obtain a buy-in from the client to inform us about the details at least two weeks in advance if it is a minor update and four weeks if it is a major one. This will provide us with ample time to make and test the necessary changes to the LLM, so that it can be deployed to production as and when the SOP update is effective. Since we are using Azure AI Search, for minor updates, which have an impact to the model output, the LLM can query the search index to find the most relevant documents dynamically at runtime. The retrieved documents are then fed as context to the model to generate responses. This approach requires no retraining of the model as the SOP updates reflect immediately by updating the search index. For major updates, we will fine-tune the model, so that it provides the most accurate and deeply integrated response patterns for our LLM.

Another point to consider is that Azure OpenAI regularly releases new model versions—to improve performance and add features. To ensure our solution stays up to date, we configure automatic updates using model versions when older versions are retired. However, it's important to be aware of potential changes in model behavior after upgrades. To minimize disruptions, we review documentation, test our application with the new version and update our code and configuration as needed.

Conclusion

This paper provides a detailed explanation of how to build a gen AI model to adjudicate claims, by leveraging Azure OpenAI service. Through comprehensive data collection, preprocessing, prompt engineering, RAG and fine-tuning, the model systematically assesses and recommends valid reasons to either pay, deny or pend the claim. Some of the key priorities for the future is to integrate the solution within the core claims adjudication system, making it platform-agnostic and exploring the possibility of applying the model with the necessary modifications to other areas in claims such as adjustments and payments. Having the model engrained within the adjudication engine will result in an increase in auto adjudication rate, thereby driving increased productivity gains. Reduced cost, enhanced scalability and interoperability are some of the outcomes that we achieve with a platform-agnostic solution. Lastly, by taking cues from this model, we can expand our gen AI footprint in the claims business—by building solutions in claims adjustments and payments.

About the author

Saptarshi Dhar is a transformation solution leader at Cognizant's Healthcare IOA business unit and is responsible for providing transformation solution and productivity commitment for all new deals. Saptarshi joined Cognizant in 2023 from Amazon, where he spent more than two years developing automated products—for ensuring a seamless customer shopping experience on the company website. Prior to Amazon, Saptarshi worked in the healthcare domain for several companies, including Sagility and Tata Consultancy Services. Saptarshi has over more than ten years of expertise in automating healthcare payer processes and developing AI/ML models.

Saptarshi Dhar

General Manager, PEx
Healthcare IOA

Reference

2024 Gartner report on top technology investments and objectives for healthcare payers (https://emt.gartnerweb.com/ngw/globalassets/en/information-technology/images/infographics/2024-cio-agenda/2024-cio-agenda-infographics/2024-cio-agenda-banking-and-investment-infographic)

2023 KPMG survey (https://kpmg.com/dk/en/home/insights/2023/10/kpmg-ceo-outlook-2023.html)

Three principal generative AI use case categories from Gartner (https://www.gartner.com/en/articles/take-this-view-to-assess-roi-for-generative-ai)

Data preprocessing (https://www.turing.com/resources/understanding-data-processing-techniques-for-llms)

Detailed solution approach with Azure OpenAI service (https://learn.microsoft.com/en-us/azure/ai-services/openai/)

Prompt engineering (https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering?tabs=chat)

RAG in Azure OpenAI (https://learn.microsoft.com/en-us/azure/search/retrieval-augmented-generation-overview)

Vector database—Azure AI search (https://medium.com/@mjtpena/choosing-the-right-azure-vector-database-d62a64f30e68)

Hybrid retrieval with semantic ranking versus vector-only search (https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/azure-ai-search-outperforming-vector-search-with-hybrid/ba-p/3929167)

Fine-tuning (https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/fine-tuning?tabs=turbo%2Cpython-new&pivots=programming-language-studio)

Model versions (https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/model-versions)

World Headquarters
300 Frank W. Burr Blvd.
Suite 36, 6th Floor
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277

European Headquarters
280 Bishopsgate
London
EC2M 4RB
England
Tel: +44 (01) 020 7297 7600

India Operations Headquarters
5/535, Okkiam Thoraipakkam,
Old Mahabalipuram Road,
Chennai 600 096
Tel: 1-800-208-6999
Fax: +91 (01) 44 4209 6060

APAC Headquarters
1 Fusionopolis Link, Level 5
NEXUS@One-North, North Tower
Singapore 138542
Tel: +65 6812 4000

WF3158451