Amazon Bedrock: “Sorry, I’m unable to assist you with this request”. Indagine e risoluzione del misterioso errore

15 Gennaio 2025 - 11 min. read

AI | Amazon Bedrock | Generative AI

"Quando hai eliminato l'impossibile, ciò che rimane, per quanto improbabile, deve essere la verità."

Sherlock Holmes

Avere un assistente (lo chiameremo Watson!) che ci aiuti a recuperare informazioni e che risponda alle nostre domande è sempre utile. Si sa, cercare all'interno di vaste documentazioni (Knowledge Base) può essere un compito molto impegnativo in termini di tempo.

Per fortuna, oggi i Large Language Model (LLM) sono qui per aiutarci... Almeno fino a che non si rifiutano di fare il loro lavoro!

Recentemente, ci siamo imbattuti nel misterioso errore “Sorry, I’m unable to assist you with this request” dato dal modello quando si esegue programmaticamente una query “retrieve&generate” utilizzando Amazon Bedrock e la libreria Python boto3.

TL;DR

Utilizzare la query boto3 Amazon Bedrock “retrieve&generate” fornendo un prompt template con il placeholder $output_format_instructions$ già valutato e creare un parser XML.

In questo modo si otterrà sempre il testo in uscita che, se sintatticamente corretto, potrà essere analizzato con il proprio parser XML personalizzato per estrarre il reale testo. Inoltre,

questo fa rimanere ancora attivo il parser XML, integrato nel metodo boto3, per cercare i riferimenti ed estrarre le citazioni dei documenti della knowledge-base.

Altrimenti, se non è allineato con la struttura XML definita, è sufficiente sanificare i tag XML per estrarre il testo in uscita.

Il problema

Ci è stato affidato il compito di implementare un sistema di ricerca basato su una Knowledge Base di una azienda farmaceutica. Abbiamo scelto Amazon Bedrock poiché la sintesi e la risposta alle query in linguaggio naturale sono facili da ottenere.

Purtroppo, le cose non vanno sempre come vorremmo; così, quando abbiamo implementato la soluzione, abbiamo scoperto che l'uso programmatico di boto3 per eseguire una query di retrieve and generate ci dava saltuariamente e in modo imprevedibile l'errore **“Sorry, I’m unable to assist you with this request”** senza informazioni aggiuntive o log.

Inutile dire che non è disponibile documentazione, né articoli su Internet, quindi abbiamo iniziato la nostra avventura per risolvere il problema. Di seguito potrete leggere come è andato il nostro viaggio e, se state avendo lo stesso problema, potrete trovare la soluzione.

Continuate a leggere!

Il nostro caso d’uso e implementazione

Facciamo una breve introduzione al caso d'uso per capire meglio il contesto e la soluzione. Sarà molto importante per capire, verifica per verifica, dove si trova il problema.

Il nostro compito era quello di implementare un sistema di ricerca per trovare facilmente i foglietti illustrativi dei farmaci tra tutta la documentazione del catalogo dei prodotti. Il testo del foglietto illustrativo doveva essere accessibile tramite API, in modo che la funzione potesse essere integrata all'interno dell'applicazione web del cliente e utilizzata dalla sua base di utenti per facilitarne l'esperienza (UX).

Il sistema complessivo può essere descritto attraverso 3 macro-aree:

- Sistema di archiviazione
- Sistema di caricamento
- Sistema di recupero

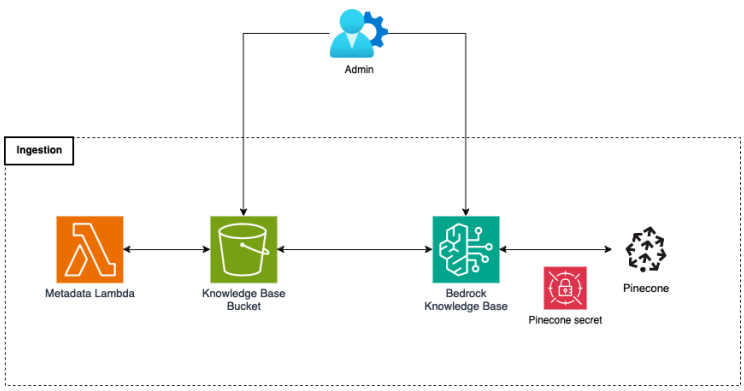
Partendo dalle fondamenta della nostra infrastruttura, il **sistema di archiviazione** supporta l'applicazione utilizzando un bucket S3, uno database vettoriale, Pinecone, e una Bedrock Knowledge Base.

Il bucket S3 contiene tutti i documenti della nostra Knowledge Base, insieme ai relativi file di metadati associati.

Il database vettoriale, o vector storage, è il cuore pulsante dell'infrastruttura di archiviazione. Contiene gli embeddings di tutti i pezzi dei nostri documenti, insieme ai metadati associati. In altre parole, questo è solo un modo complicato per dire che contiene l'intero testo della Knowledge base, diviso in pezzi (chunks), ognuno dei quali trasformato con una rappresentazione vettoriale (embedding) che è molto efficace ed efficiente per le attività di ricerca e recupero, poiché codifica le informazioni semantiche del testo al suo interno. Date le dimensioni complessive dei documenti, abbiamo scelto Pinecone per la nostra soluzione poiché si integra facilmente con i pezzi della nostra infrastruttura AWS ed è molto efficiente dal punto di vista dei costi per il caso d'uso.

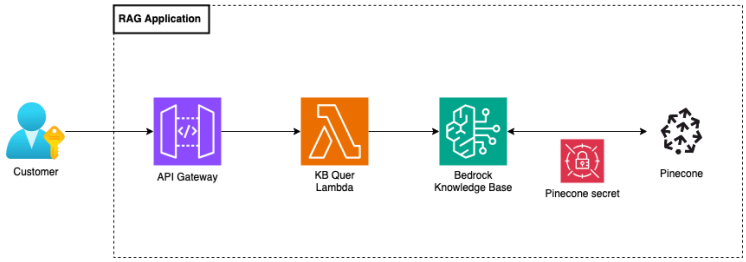
I documenti del bucket S3 vengono elaborati e trasferiti nel database vettoriale utilizzando la Knowledge Base di Bedrock. Si tratta di un elemento infrastrutturale che prepara e carica i documenti da varie fonti di dati configurate all'interno del database vettoriale. Inoltre, la Knowledge Base di Bedrock gestisce anche il recupero dei documenti significativi in base a un'interrogazione dell'utente, elemento chiave della soluzione.

Il sistema di caricamento automatizza l'elaborazione e il popolamento del database vettoriale. Quando un utente amministratore carica un nuovo documento all'interno del bucket S3, viene avviata una funzione Lambda che genera un file di metadati associato al documento di origine. L'amministratore sincronizzerà i nuovi documenti della knowledge base nel database vettoriale Pinecone utilizzando la già citata funzione della Bedrock Knowledge Base.



Infine, **il sistema di recupero** è composto da un API Gateway che gestisce le richieste provenienti dall'applicazione del cliente utilizzando una funzione Lambda. La funzione

prende in input una domanda dell'utente e interroga la base di conoscenza, utilizzando la chiamata API retrieve&generate della libreria Python boto3, per reperire il testo del foglietto illustrativo richiesto. La chiamata API retrieve&generate utilizza il LLM Anthropic Claude 3 Sonnet, il migliore per il nostro caso d'uso al momento del progetto.



Per completare il contesto, abbiamo caricato l'intera documentazione del catalogo, circa 70 documenti, all'interno del bucket S3 e abbiamo avviato il processo di sincronizzazione. Una volta terminato, abbiamo iniziato a testare la soluzione e ci siamo imbattuti nel famigerato errore: “Sorry, I'm unable to assist you with this request”, un codice di stato 200 OK e nessun altro indizio.

Alla ricerca dell’errore

La ricerca della vera fonte di questo errore è stata molto dura. La strada era piena di trappole: pericoli logici che cercavano di portarci fuori strada. Rivediamo il processo di ricerca in modo più strutturato e ordinato passo dopo passo:

Step 1: Generare l'errore

Passo molto semplice e basilare: abbiamo provato la soluzione con vari prompt e domande degli utenti per capire se ci fossero pattern colpevoli dell'errore.

L'errore sembrava però verificarsi in momenti casuali. Per risolvere il problema, abbiamo fissato alcune variabili, generato alcune ipotesi ed eseguito ulteriori test.

Abbiamo testato alcuni prompt per la parte di generazione e abbiamo definito il migliore da utilizzare nella successiva serie di test. Una volta che la variabile prompt è stata definita, abbiamo potuto creare alcune ipotesi per capire se il problema si nascondesse nei documenti. Definiamole:

- **Ipotesi 1:** il problema è causato da richieste degli utenti su documenti mancanti/non esistenti?
- **Ipotesi 2**: il problema è causato da richieste degli utenti su documenti specifici?

Step 2: Test delle ipotesi sui documenti

Abbiamo iniziato a testare la prima ipotesi ponendo una serie di domande su farmaci specifici e alcune su farmaci inesistenti. L'esito del test ha invalidato l'ipotesi: il testo in uscita per i farmaci inesistenti a volte era corretto, a volte affetto da errore.

Successivamente, abbiamo eseguito un test simile per la seconda ipotesi, raccogliendo le domande precedenti degli utenti, insieme ai loro output. Abbiamo eseguito il test comprendendo che il problema sembra essere legato a documenti specifici, mentre non c'era correlazione tra i documenti “difettosi”.

Qui il mistero si è infittito. L'errore era causato da un insieme di documenti specifici, molto diversi tra loro. Inoltre, alcuni documenti difettosi erano molto simili a quelli corretti!

Stavamo comunque facendo dei passi avanti per avvicinarci alla soluzione. Il passo successivo è stato capire cosa rendesse un documento “difettoso”. Abbiamo limitato l'entropia restringendo la nostra Knowledge base solo a quest'ultima tipologia di documenti e abbiamo eseguito la chiamata API “retrieve” per vedere le differenze nei pezzi recuperati tra i documenti buoni e quelli cattivi.

Step 3: Ritorno alle origini, pochi documenti, solo recupero

Abbiamo creato un ambiente isolato che riflettesse la soluzione e abbiamo caricato nella Knowledge base solo i documenti dell'ultimo test, cercando di capire le differenze tra loro.

Abbiamo creato un notebook per eseguire alcune analisi sul nostro database vettoriale e ricavarne alcuni spunti.

In primo luogo, abbiamo esaminato il numero di chunk all'interno del database vettoriale: Il numero di chunk era paragonabile al numero di documenti originali. Ciò non sorprende, in quanto i foglietti illustrativi dei farmaci sono documenti di piccole dimensioni e quasi tutti possono essere contenuti nella dimensione dei chunk.

Quindi, abbiamo eseguito alcune operazioni di recupero utilizzando la chiamata API boto3 retrieve e le domande del nostro ultimo test. Ed ecco un risultato inaspettato: tutti i documenti erano stati recuperati!

Con queste informazioni, il problema doveva trovarsi nella parte di generazione della chiamata API boto3 retrieve&generate. Qui abbiamo solo due variabili: il prompt e le domande.

Ci stiamo avvicinando alla soluzione, ma come arrivarci? Ovviamente... più log!

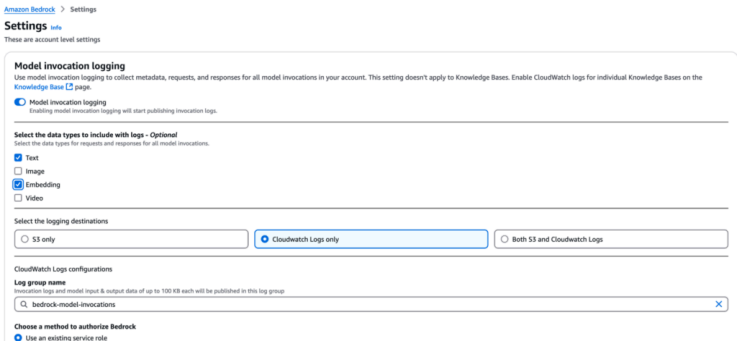
Step 4: Bedrock Logging

Avevamo bisogno di ulteriori informazioni per capire meglio come funzionasse la chiamata API retrieve&generate delle libreria Python boto3, come venissero gestiti i prompt e se domande specifiche potessero innescare qualche condizione inaspettata.

Per ottenere maggiori dettagli, abbiamo attivato il logging di Bedrock.

Per farlo, serve entrare nelle Impostazioni, sotto la voce Configurazioni di Bedrock, e attivare il logging delle invocazioni dei modelli.

È necessario specificare per quali tipi di modelli si desidera registrare le invocazioni e specificare un ruolo IAM per salvare i log alle destinazioni proposte: CloudWatch, S3 o entrambi. Poiché si trattava di documenti di tipo testuale, abbiamo registrato solo gli embedding e le invocazioni dei modelli di testo. In questo caso, i log di CloudWatch sono stati sufficienti. Ecco un esempio di configurazione:



Una nota rapida, ma importante: abilitando il logging, ogni singola invocazione di un modello viene registrata. Questi log possono crescere molto velocemente e, con essi, anche la vostra bolletta! Per evitare fatture inaspettate alla fine del mese, attivate questi log solo quando ne avete veramente bisogno.

Abbiamo rieseguito alcune chiamate retrieve&generate per capire cosa succedesse al prompt, alle domande e all'output.

Cosa abbiamo trovato (il tassello mancante)

Dai log delle invocazioni, vediamo finalmente la luce!

Il problema era all'interno del prompt e riguardava in particolare l'uso del placeholder $output_format_instructions$.

Il segnaposto costringe il LLM, Claude 3 Sonnet in questo caso, a produrre una specifica struttura XML. Questa struttura è necessaria per trovare le citazioni all'interno del testo della Knowledge base. Come afferma la documentazione AWS: “Senza questo segnaposto, la risposta non conterrà citazioni”.

Per chi fosse curioso, nella nostra configurazione il placeholder viene tradotto con questo testo:

```
If you reference information from a search result within your answer, you must include a citation to source where the information was found. Each result has a corresponding source ID that you should reference.

Note that <sources> may contain multiple <source> if you include information from multiple results in your answer.

Do NOT directly quote the <search_results> in your answer. Your job is to answer the user's question as concisely as possible.

You must output your answer in the following format. Pay attention and follow the formatting and spacing exactly:
<answer>
        <answer_part>
        <text>first answer text</text>
        <sources>
                <source>source ID</source>
        </sources>
        </answer_part>
        <answer_part>
        <text>second answer text</text>
        <sources>
                <source>source ID</source>
        </sources>
        </answer_part>
        <answer_part>
        <text>n-th answer text</text>
        <sources>
                <source>source ID</source>
        </sources>
        </answer_part>
</answer>

Assistant:
```

Il problema si manifesta quando il testo di output prodotto dal LLM non è conforme alla struttura XML prevista. Quando ciò accade, il codice interno del metodo retrieve&generate

di boto3 solleva un'eccezione che viene gestita come abbiamo visto in precedenza: un codice di stato 200 OK e il testo di output “Sorry, I'm unable to assist you with this request”.

La nostra ipotesi è che ci sia un parser XML che viene attivato ogni volta che il prompt template contiene il placeholder **$output_format_instructions$**. Questo parser è responsabile dell'estrazione del testo di output dalla struttura XML e delle relative citazioni. Quando il testo di output del LLM non è conforme alla struttura XML, il parser non è in grado di svolgere il suo lavoro e solleva un'eccezione, gestita come indicato sopra.

La soluzione

Ora che abbiamo trovato il colpevole, risolviamo il problema!

Abbiamo modificato il prompt template, sostituendo manualmente il placeholder $output_format_instructions$ con il testo precedentemente citato. Così, siamo stati in grado di ottenere sempre la risposta del LLM, ma fortunatamente o sfortunatamente, la maggior parte delle volte, è giustamente incapsulata all'interno della struttura XML.

Qui notiamo una caratteristica particolare: anche se il segnaposto $output_format_instructions$ non è incluso nel prompt template, il parser di boto3 ottiene comunque le citazioni se l'output LLM è formattato correttamente con la struttura XML definita. Questo indica che l'errore non è causato dal semplice parsing dell'output XML del LLM, bensì dall'estrazione del testo di output dall'XML.

Abbiamo scritto alcune righe di codice per sviluppare un parser/sanitizzatore XML molto semplice e abbiamo preso due piccioni con una fava: ottenere il testo in uscita e le relative citazioni.

Dopo aver eseguito alcuni test, il sistema è ora stabile e siamo riusciti a integrare la soluzione con l'applicazione web del cliente.

Conclusioni

In questo articolo, abbiamo illustrato l'interazione con una Knowledge Base Bedrock. Abbiamo esplorato l'integrazione con un database vettoriale e alcune intuizioni sulla distribuzione dei documenti. Abbiamo analizzato la chiamata API retrieve&generate della libreria boto3 e la sua interazione con il prompt template, nonché il suo meccanismo di funzionamento interno e abbiamo mostrato un modo per approfondire l'argomento utilizzando la funzione di registrazione delle invocazioni dei modelli di Bedrock. Tutto

questo viaggio e finalmente abbiamo risolto il famigerato testo di output “Sorry, I'm unable to assist you with this request”. Eureka!

Vi è mai capitato di affrontare un problema simile? A volte far luce sull'ignoto può insegnarci molto sui dettagli interni e sul funzionamento di un servizio. Raccontateci il vostro viaggio nei commenti!

About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

Matteo Goretti

DevOps Engineer @ beSharp. Appassionato di Cloud Computing e Intelligenza Artificiale, in particolare, Machine Learning e Deep Learning. Amo il trekking e la natura in generale. Mi rilasso con la mia chitarra, giocando ai videogames o guardando serie TV.
