Using Impala AI function in Data Warehouse Public Cloud (Preview)

Date published: 2024-07-26
Date modified: 2024-07-26

Legal Notice

Contents

Advantages and use cases of Impala AI functions

Cloudera Data Warehouse allows you to use Impala's ai_generate_text function to access Large Language Models (LLMs) in SQL queries. This function enables you to input a prompt, retrieve the LLM response, and include it in results. You can create custom UDFs for complex tasks like sentiment analysis and translation.

Note: *This feature is in technical preview and not recommended for production deployments. Cloudera recommends that you try this feature in test and development environments.*

Use LLMs directly in SQL with Impala's ai_generate_text function

Impala introduces a built-in AI function called `ai_generate_text` that enables direct access to and utilization of Large Language Models (LLMs) in SQL queries. With this function, you can input a prompt, which may include data. The function communicates with a supported LLM endpoint, sends the prompt, retrieves the response, and includes it in the query result.

Alternatively, seamlessly integrate LLM intelligence into your Impala workflow by creating custom User Defined Functions (UDFs) on top of `ai_generate_text`. This allows you to use concise SQL statements for sending prompts to an LLM and receiving responses. You can define UDFs for complex tasks like sentiment analysis, language translation, and generative contextual analysis.

For examples of AI built-in function arguments and how to create and use custom UDFs with the built-in AI function, see *Overview of AI function syntax and arguments.*

Advantages of using AI functions

- Simplified Workflow: Eliminates the necessity for setting up intricate data pipelines.
- No ML Expertise Required: No specialized machine learning skills are needed.

- Swift Decision-Making: Enables faster insights on the data, facilitating critical business decisions by using in-database function calls.
- Integrated Functionality: Requires no external applications, as it is a built-in feature in Data Warehouse.

List of possible use cases

Here are some practical applications of using AI models with the function:

- Sentiment Analysis: Use the AI model to examine customer reviews for a product and identify their sentiment as positive, negative, or neutral.
- Language Translation: Translate product reviews written in different languages to understand customer feedback from various regions.
- Generative Contextual Analysis: Generate detailed reports and insights on various topics based on provided data.

Overview of AI function syntax and arguments

Learn how you can use the ai_generate_text functions to connect to OpenAI or Azure OpenAI endpoints and integrate AI models into your Impala SQL queries.

Syntax for AI built-in function arguments

The following example of a built-in AI function demonstrates the use of the OpenAI API as a large language model. Currently, OpenAI's public endpoint and Azure OpenAI endpoints are supported.

AI_GENERATE_TEXT_DEFAULT

Syntax

```
ai_generate_text_default(prompt)
```

AI_GENERATE_TEXT

Syntax

```
ai_generate_text(ai_endpoint, prompt, ai_model,
ai_api_key_jceks_secret, additional_params)
```

The `ai_generate_text` function uses the values you provide as an argument in the function for `ai_ endpoint`, `ai_model`, and `ai_api_key_jceks_secret`. If any of them are left empty or NULL, the function uses the default values set at the instance level instead.

When using the `ai_generate_text_default` function, make sure to set all parameters (`ai_ endpoint`, `ai_model`, and `ai_api_key_jceks_secret`) in the coordinator/executor flagfiles with appropriate values.

Key parameters for using the AI model

- ai_endpoint: The endpoint for the model API that is being interfaced with, supports services like OpenAI and Azure OpenAI Service, for example, https://api.openai.com/v1/chat/completions.
- prompt: The text you submit to the AI model to generate a response.
- ai_model: The specific model name you want to use within the desired API, for example, gpt-3.5-turbo.
- ai_api_key_jceks_secret: The key name for the JCEKS secret that contains your API key for the AI API you are using. You need a JCEKS keystore containing the specified JCEKS secret referenced in ai_api_key_jceks_secret. To configure the keystore file location, see *Configuring the JCEKS keystore for Impala AI integration*.
- additional_params: Additional parameters that the AI API offers that is provided to the built-in function as a JSON object.

Examples of AI functions and UDFs

Learn to use the ai_generate_text_default function for tasks like analyzing Amazon book reviews. Also, see how to create custom UDFs to make AI text analysis easier and more efficient, enhancing your data workflows and insights.

Examples of using the built-in AI function

The following example lists the steps needed to turn a prompt into a custom SQL function using just the built-in function `ai_generate_text_default`.

```
> select ai_generate_text_default('hello');

Response:

Hello! How can I assist you today?
```

In the below example, a query is sent to the Amazon book reviews database for the book titled Artificial Superintelligence. The large language model (LLM) is prompted to classify the sentiment as positive, neutral, or negative.

```
> select customer_id, star_rating,
ai_generate_text_default(CONCAT('Classify the following review as
positive, neutral, or negative', and only include the uncapitalized
category in the response: ', review_body)) AS review_analysis,
review_body from amazon_book_reviews where product_title='Artificial
Superintelligence' order by customer_id LIMIT 1;
```

```
Response:

+--+-----------+-----------+---------------+------------------+
|  |customer_id |star_rating |review_analysis |review_body       |
+--+-----------+-----------+---------------+------------------+
|1 |4343565     | 5          |positive        |What is this book |
|  |            |            |                |all about ……      |
+--+-----------+-----------+---------------+------------------+
```

Examples of creating and using custom UDFs along with the built-in AI function

Instead of writing the prompts in a SQL query, you can build an UDF with your intended prompt. Once you build your custom UDF with the prompt, you can pass that prompt into the ai_generate_text_default built-in Impala function.

Example: Classify input customer reviews

The following UDF uses the Amazon book reviews database as the input and requests the LLM to classify the sentiment.

Classify input customer reviews:

```
IMPALA_UDF_EXPORT
```

```
StringVal ClassifyReviewsDefault(FunctionContext* context, const
StringVal& input) {

 std::string request =

     std::string("Classify the following review as positive, neutral,
or negative")

     + std::string(" and only include the uncapitalized category in
the response: ")

     + std::string(reinterpret_cast<const char*>(input.ptr),
input.len);

 StringVal prompt(request.c_str());

 return context->Functions()->ai_generate_text_default(context,
prompt);

}
```

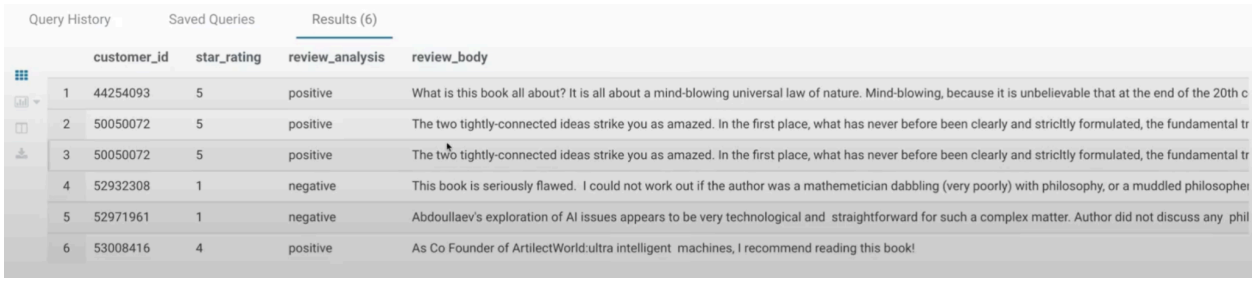
Now you can define these prompt building UDF and build them in Impala. Once you have them running, you can query your data sets using them.

Creating `analyze_reviews` function:

```
> CREATE FUNCTION analyze_reviews(STRING)

RETURNS STRING

LOCATION 's3a://dw-..............'

SYMBOL='ClassifyReviews'
```

Using SELECT query for Sentiment analysis to classify Amazon book reviews

```
> SELECT customer_id, star_rating, analyze_reviews(review_body) AS
review_analysis, review_body from amazon_book_reviews where
product_title='Artificial Superintelligence' order by customer_id;
```



Configuring the JCEKS keystore for Impala AI integration

Learn to configure the JCEKS keystore for AI functions in Impala, including setting the keystore file location and updating Cloudera Data Warehouse settings. This ensures secure storage and management of API keys, facilitating a secure and efficient AI integration within data workflows.

1. Log in to the Cloudera Data Warehouse service as an administrator.
2. Go to **Impala Virtual Warehouse→→ Edit→Configurations→Impala coordinator** and select `hadoop-core-site-default-warehouse` from the **Configuration files** drop-down list.
3. Set the value of the `hadoop.security.credential.provider.path` property to **`[***FILE PATH***]`**.localjceks.
4. Go to the **Impala executor** tab and select `hadoop-core-site-default-warehouse` from the **Configuration files** drop-down list.
5. Set the value of the `hadoop.security.credential.provider.path` property to **`[***FILE PATH***]`**.localjceks.
6. Restart the Impala Virtual Warehouse.