perplexity

How to fix "Gemini throwing bad request error when passing human input" with crewai

CrewAI is a powerful framework for building agent-based systems that can leverage various LLM providers, including Google's Gemini models. However, users frequently encounter a specific error when attempting to implement human input functionality with Gemini models in their CrewAI applications. This comprehensive report examines the "Bad Request" error that occurs when passing human input to Gemini models and provides detailed solutions to resolve these issues.

Understanding the Gemini Bad Request Error in CrewAI

When implementing human input functionality with Google's Gemini models in CrewAI, users often encounter a specific error pattern. The error typically manifests as follows:

```
ERROR: LiteLLM call failed: litellm.BadRequestError: VertexAIException BadRequestError -
"error": {
"code": 400,
"message": "* GenerateContentRequest.contents: contents is not specified\n",
"status": "INVALID_ARGUMENT"
}
```

This error occurs even when users have explicitly set `human_input=True` in their agent or task configuration[1]. The primary issue stems from how CrewAI interfaces with the Gemini API through LiteLLM, which is the underlying library handling API calls to various LLM providers. When CrewAI attempts to process human feedback through Gemini, the system fails to properly format the request, resulting in a missing "contents" field in the GenerateContentRequest object sent to the Gemini API[1].

Root Causes of the Bad Request Error

The error fundamentally arises from compatibility issues between CrewAI's human input processing mechanism and the specific requirements of the Gemini API. Several factors contribute to this problem:

Incorrect LLM Provider Configuration

One common cause is improper specification of the LLM provider when initializing the model. When using Gemini with CrewAI, the model name must follow a specific format: `gemini/<LLM name>`[2]. Failing to specify the provider correctly causes LiteLLM to throw an error as it cannot determine which API to use for processing the request[2].

Limitations in Human Input Implementation

CrewAI's human input functionality has inherent limitations that affect how it works with different models. Users report that human input is often only requested at the end of a task rather than iteratively throughout the process[3]. This implementation can cause particular problems with Gemini models, which may expect differently structured requests for human input processing[3].

API Structure Mismatch

The Gemini API expects a specific structure for its requests, particularly when handling interactive elements like human input. When CrewAI attempts to classify or process human feedback through Gemini, there's a mismatch between what CrewAI sends and what the Gemini API expects in terms of the "contents" field[1].

Solutions for Fixing the Bad Request Error

Based on reported issues and successful implementations, here are detailed solutions to resolve the Gemini bad request error when handling human input in CrewAI:

Proper LLM Configuration Using CrewAI's LLM Class

When working with Gemini models in CrewAI, it's crucial to use the appropriate configuration through the CrewAI LLM class. This class properly leverages LiteLLM in the background to handle provider-specific requirements[2]. The correct implementation looks like this:

```
from crewai import LLM

llm = LLM(
    model="gemini/gemini-1.5-pro",  # Correct format with provider prefix
    api_key=GEMINI_API_KEY,
    temperature=0.7
)

# Then use this LLM instance when creating agents
agent = Agent(
    role="Content Creator",
    goal="Create engaging content",
    backstory="You're an expert content creator",
    llm=llm,
    allow_delegation=False,
    verbose=True
)
```

This implementation ensures that LiteLLM correctly identifies Gemini as the provider and formats requests appropriately[2].

Updating CrewAI and LiteLLM Versions

Many issues with Gemini integration in CrewAI stem from version incompatibilities. Ensuring you have the latest versions of both CrewAI and LiteLLM can resolve many of these problems, as developers frequently push fixes for these specific integration issues[4] [5]. Update your packages using:

```
pip install --upgrade crewai
pip install --upgrade litellm
```

Alternative Human Input Implementation

Since the native human input functionality in CrewAI can be problematic with Gemini, consider implementing a custom solution for gathering human input. Instead of relying on CrewAI's built-in human_input parameter, you can:

1. Create a custom tool that handles input collection
2. Use Python's built-in input() function at strategic points in your workflow
3. Implement a feedback loop mechanism outside of the CrewAI task structure[6]

This approach bypasses the problematic internal processing that leads to the bad request error.

Best Practices for Human Input with Gemini in CrewAI

To ensure reliable human input processing when using Gemini with CrewAI, follow these best practices:

Implement Robust Error Handling

Given the known issues with human input in CrewAI, especially with Gemini models, implementing comprehensive error handling is essential. This helps identify where problems occur and provides graceful fallbacks:

```
try:
    # Attempt to process human input with CrewAI
    result = crew.kickoff(inputs={"topic": "Artificial Intelligence"})
except Exception as e:
    print(f"Error during CrewAI execution: {e}")
    # Implement fallback mechanism for human input
    manual_input = input("Please provide your input manually: ")
    # Process the manual_input accordingly
```

Structure Tasks to Minimize Human Input Dependency

Since human input functionality can be problematic, structure your CrewAI workflows to minimize dependence on this feature. Consider designing your agent system to batch human input requirements or to process them at specific checkpoints rather than requiring continuous interactive feedback[3] [6].

Verify API Keys and Authentication

Authentication issues can sometimes manifest in ways similar to the bad request error. Always verify that your Gemini API key is correct and properly configured in your environment variables or configuration files[4]. Some users have reported that what appeared to be content-related errors were actually authentication issues in disguise.

Troubleshooting Persistent Issues

If you continue experiencing the bad request error after implementing the solutions above, consider these additional troubleshooting steps:

Enable Verbose Logging

Turn on verbose logging for both CrewAI and LiteLLM to get more detailed information about what's happening during the API calls:

```
import litellm
litellm.set_verbose = True

# And when creating agents:
agent = Agent(
    # other parameters
    verbose=True
)
```

The error messages often suggest this step: "LiteLLM.Info: If you need to debug this error, use `litellm.set_verbose=True`"[1].

Examine the Request Payload

If possible, capture and examine the actual request payload being sent to the Gemini API. This can help identify exactly what's missing or malformed in the request. You might need to modify CrewAI's source code or use network monitoring tools to capture this information.

Use Alternative Models for Human Input Processing

If the issue persists specifically with Gemini, consider using a different model for tasks that require human input processing. For example, you could use OpenAI's models for tasks requiring human interaction while still using Gemini for other tasks within your CrewAI workflow[6].

Conclusion

The "Bad Request" error when passing human input to Gemini models in CrewAI stems from compatibility issues between CrewAI's human input implementation and the Gemini API's requirements. By properly configuring the LLM provider, updating dependencies, implementing robust error handling, and considering alternative approaches to human input collection, most users can resolve these issues.

As both CrewAI and Google's Gemini API continue to evolve, we can expect improved compatibility and more seamless integration in the future. In the meantime, the workarounds and solutions outlined in this report should help developers successfully implement human input functionality with Gemini models in their CrewAI applications.

⁂

1. https://community.deeplearning.ai/t/gemini-throwing-bad-request-error-when-passing-human-input/754637
2. https://stackoverflow.com/questions/79111773/litellm-badrequesterror-llm-provider-not-provided-pass-in-the-llm-provider-you
3. https://www.reddit.com/r/crewai/comments/1dbwq8k/has_anyone_gotten_the_human_input_to_work_properly/
4. https://community.crewai.com/t/issue-with-invalid-api-key-in-crewai-agent-for-gemini-api/4010
5. https://community.crewai.com/t/gemini-has-stopped-working-again/2209
6. https://stackoverflow.com/questions/78738674/trouble-handling-user-input-and-script-functionality-issue-in-crewai-and-langcha