Google Cloud Platform Vertex AI PaLM

Note: This is seperate from the Google PaLM integration. Google has chosen to offer an enterprise version of PaLM through GCP, and this supports the models made available through there.

PaLM API on Vertex AI is a Preview offering, subject to the Pre-GA Offerings Terms of the GCP Service Specific Terms.

Pre-GA products and features may have limited support, and changes to pre-GA products and features may not be compatible with other pre-GA versions. For more information, see the launch stage descriptions. Further, by using PaLM API on Vertex AI, you agree to the Generative AI Preview terms and conditions (Preview Terms).

For PaLM API on Vertex AI, you can process personal data as outlined in the Cloud Data Processing Addendum, subject to applicable restrictions and obligations in the Agreement (as defined in the Preview Terms).

To use Vertex AI PaLM you must have the `google-cloud-aiplatform` Python package installed and either:

- Have credentials configured for your environment (gcloud, workload identity, etc...)
- Store the path to a service account JSON file as the GOOGLE_APPLICATION_CREDENTIALS environment variable

This codebase uses the `google.auth` library which first looks for the application credentials variable mentioned above, and then looks for system-level auth.

For more information, see:

- https://cloud.google.com/docs/authentication/application-default-credentials#GAC
- https://googleapis.dev/python/google-auth/latest/reference/google.auth.html#module-google.auth

```
#!pip install google-cloud-aiplatform
```

```
from langchain.chat_models import ChatVertexAI
from langchain.prompts.chat import (
    ChatPromptTemplate,
    SystemMessagePromptTemplate,
    HumanMessagePromptTemplate,
)
from langchain.schema import HumanMessage, SystemMessage
```

```
chat = ChatVertexAI()
```

```
messages = [
    SystemMessage(
        content="You are a helpful assistant that translates English to French."
    ),
    HumanMessage(
        content="Translate this sentence from English to French. I love programming."
    ),
]
chat(messages)
```

```
    AIMessage(content='Sure, here is the translation of the sentence "I love programming" from English to French:\n\nJ\'aime programmer.', additional_kwargs={}, example=False)
```

You can make use of templating by using a `MessagePromptTemplate`. You can build a `ChatPromptTemplate` from one or more `MessagePromptTemplates`. You can use `ChatPromptTemplate`'s `format_prompt` -- this returns a `PromptValue`, which you can convert to a string or Message object, depending on whether you want to use the formatted value as input to an llm or chat model.

For convenience, there is a `from_template` method exposed on the template. If you were to use this template, this is what it would look like:

```
template = (
    "You are a helpful assistant that translates {input_language} to {output_language}."
)
system_message_prompt = SystemMessagePromptTemplate.from_template(template)
human_template = "{text}"
human_message_prompt = HumanMessagePromptTemplate.from_template(human_template)
```

```
chat_prompt = ChatPromptTemplate.from_messages(
    [system_message_prompt, human_message_prompt]
)

# get a chat completion from the formatted messages
chat(
    chat_prompt.format_prompt(
        input_language="English", output_language="French", text="I love programming."
    ).to_messages()
)
```

```
    AIMessage(content='Sure, here is the translation of "I love programming" in French:\n\nJ\'aime
programmer.', additional_kwargs={}, example=False)
```

You can now leverage the Codey API for code chat within Vertex AI. The model name is:

- codechat-bison: for code assistance

```
chat = ChatVertexAI(model_name="codechat-bison")
```

```
messages = [
    HumanMessage(content="How do I create a python function to identify all prime numbers?")
]
chat(messages)
```

```
    AIMessage(content='The following Python function can be used to identify all prime numbers up to a given integer:\n\n```\ndef is_prime(n):\n  """\n  Determines whether the given integer is prime.\n\n  Args:\n    n: The integer to be tested for primality.\n\n  Returns:\n    True if n is prime, False otherwise.\n  """\n\n  # Check if n is divisible by 2.\n  if n % 2 == 0:\n    return False\n\n  # Check if n is divisible by any integer from 3 to the square root', additional_kwargs={}, example=False)
```