Modules Model I/O Language models LLMs Streaming

Streaming

Some LLMs provide a streaming response. This means that instead of waiting for the entire response to be returned, you can start processing it as soon as it's available. This is useful if you want to display the response to the user as it's being generated, or if you want to process the response as it's being generated.

Currently, we support streaming for a broad range of LLM implementations, including but not limited to `OpenAI`, `ChatOpenAI`, `ChatAnthropic`, `Hugging Face Text Generation Inference`, and `Replicate`. This feature has been expanded to accommodate most of the models. To utilize streaming, use a `CallbackHandler` that implements `on_llm_new_token`. In this example, we are using `StreamingStdOutCallbackHandler`.

```
from langchain.llms import OpenAI
from langchain.callbacks.streaming_stdout import
StreamingStdOutCallbackHandler


llm = OpenAI(streaming=True, callbacks=[StreamingStdOutCallbackHandler()],
temperature=0)
resp = llm("Write me a song about sparkling water.")
```

```
    Verse 1
    I'm sippin' on sparkling water,
    It's so refreshing and light,
    It's the perfect way to quench my thirst
    On a hot summer night.

    Chorus
    Sparkling water, sparkling water,
    It's the best way to stay hydrated,
    It's so crisp and so clean,
    It's the perfect way to stay refreshed.

    Verse 2
    I'm sippin' on sparkling water,
    It's so bubbly and bright,
    It's the perfect way to cool me down
    On a hot summer night.

    Chorus
    Sparkling water, sparkling water,
    It's the best way to stay hydrated,
    It's so crisp and so clean,
    It's the perfect way to stay refreshed.

    Verse 3
    I'm sippin' on sparkling water,
    It's so light and so clear,
    It's the perfect way to keep me cool
    On a hot summer night.

    Chorus
    Sparkling water, sparkling water,
    It's the best way to stay hydrated,
    It's so crisp and so clean,
    It's the perfect way to stay refreshed.
```

We still have access to the end `LLMResult` if using `generate`. However, `token_usage` is not currently supported for streaming.

```
llm.generate(["Tell me a joke."])
```

```
    Q: What did the fish say when it hit the wall?
    A: Dam!



    LLMResult(generations=[[Generation(text='\n\nQ: What did the fish say when it hit the wall?\nA: Dam!', generation_info={'finish_reason': 'stop', 'logprobs': None})]], llm_output={'token_usage': {}, 'model_name': 'text-davinci-003'})
```