LLMs

Large Language Models (LLMs) are a core component of LangChain. LangChain does not serve it's own LLMs, but rather provides a standard interface for interacting with many different LLMs.

For more detailed documentation check out our:

- **How-to guides**: Walkthroughs of core functionality, like streaming, async, etc.
- **Integrations**: How to use different LLM providers (OpenAI, Anthropic, etc.)

Get started

There are lots of LLM providers (OpenAI, Cohere, Hugging Face, etc) - the `LLM` class is designed to provide a standard interface for all of them.

In this walkthrough we'll work with an OpenAI LLM wrapper, although the functionalities highlighted are generic for all LLM types.

Setup

To start we'll need to install the OpenAI Python package:

```
pip install openai
```

Accessing the API requires an API key, which you can get by creating an account and heading here. Once we have a key we'll want to set it as an environment variable by running:

```
export OPENAI_API_KEY="..."
```

If you'd prefer not to set an environment variable you can pass the key in directly via the `openai_api_key` named parameter when initiating the OpenAI LLM class:

```
from langchain.llms import OpenAI

llm = OpenAI(openai_api_key="...")
```

otherwise you can initialize without any params:

```
from langchain.llms import OpenAI

llm = OpenAI()
```

`__call__`: string in -> string out

The simplest way to use an LLM is a callable: pass in a string, get a string completion.

```
llm("Tell me a joke")
```

```
    'Why did the chicken cross the road?\n\nTo get to the other side.'
```

`generate`: batch calls, richer outputs

`generate` lets you can call the model with a list of strings, getting back a more complete response than just the text. This complete response can includes things like multiple top responses and other LLM provider-specific information:

```
llm_result = llm.generate(["Tell me a joke", "Tell me a poem"]*15)
```

```
len(llm_result.generations)
```

```
30
```

```
llm_result.generations[0]
```

```
[Generation(text='\n\nWhy did the chicken cross the road?\n\nTo get to the other side!'),
 Generation(text='\n\nWhy did the chicken cross the road?\n\nTo get to the other side.')]
```

```
llm_result.generations[-1]
```

```
[Generation(text="\n\nWhat if love neverspeech\n\nWhat if love never ended\n\nWhat if love was only a feeling\n\nI'll never know this love\n\nIt's not a feeling\n\nBut it's what we have for each other\n\nWe just know that love is something strong\n\nAnd we can't help but be happy\n\nWe just feel what love is for us\n\nAnd we love each other with all our heart\n\nWe just don't know how\n\nHow it will go\n\nBut we know that love is something strong\n\nAnd we'll always have each other\n\nIn our lives."),
```

```
    Generation(text='\n\nOnce upon a time\n\nThere was a love so pure and true\n\nIt lasted for centuries\n\nAnd never became stale or dry\n\nIt was moving and alive\n\nAnd the heart of the love-ick\n\nIs still beating strong and true.')]
```

You can also access provider specific information that is returned. This information is NOT standardized across providers.

```
llm_result.llm_output
```

```
{'token_usage': {'completion_tokens': 3903,
  'total_tokens': 4023,
  'prompt_tokens': 120}}
```