Caching

LangChain provides an optional caching layer for Chat Models. This is useful for two reasons:

It can save you money by reducing the number of API calls you make to the LLM provider, if you're often requesting the same completion multiple times. It can speed up your application by reducing the number of API calls you make to the LLM provider.

```
import langchain
from langchain.chat_models import ChatOpenAI

llm = ChatOpenAI()
```

In Memory Cache

```
from langchain.cache import InMemoryCache
langchain.llm_cache = InMemoryCache()

# The first time, it is not yet in cache, so it should take longer
llm.predict("Tell me a joke")
```

```
    CPU times: user 35.9 ms, sys: 28.6 ms, total: 64.6 ms
    Wall time: 4.83 s


    "\n\nWhy couldn't the bicycle stand up by itself? It was...two
tired!"
```

```
# The second time it is, so it goes faster
llm.predict("Tell me a joke")
```

```
    CPU times: user 238 µs, sys: 143 µs, total: 381 µs
    Wall time: 1.76 ms
```

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

SQLite Cache

```
rm .langchain.db
```

```
# We can do the same thing with a SQLite cache
from langchain.cache import SQLiteCache
langchain.llm_cache = SQLiteCache(database_path=".langchain.db")
```

```
# The first time, it is not yet in cache, so it should take longer
llm.predict("Tell me a joke")
```

```
    CPU times: user 17 ms, sys: 9.76 ms, total: 26.7 ms
    Wall time: 825 ms


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

```
# The second time it is, so it goes faster
llm.predict("Tell me a joke")
```

```
    CPU times: user 2.46 ms, sys: 1.23 ms, total: 3.7 ms
    Wall time: 2.67 ms


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