OpenAI Adapter

A lot of people get started with OpenAI but want to explore other models. LangChain's integrations with many model providers make this easy to do so. While LangChain has it's own message and model APIs, we've also made it as easy as possible to explore other models by exposing an adapter to adapt LangChain models to the OpenAI api.

At the moment this only deals with output and does not return other information (token counts, stop reasons, etc).

```
import openai
from langchain.adapters import openai as lc_openai
```

API Reference:

- openai from `langchain.adapters`

ChatCompletion.create

```
messages = [{"role": "user", "content": "hi"}]
```

Original OpenAI call

```
result = openai.ChatCompletion.create(
    messages=messages,
    model="gpt-3.5-turbo",
    temperature=0
)
```

```
result["choices"][0]['message'].to_dict_recursive()
```

```
    {'role': 'assistant', 'content': 'Hello! How can I assist you
today?'}
```

LangChain OpenAI wrapper call

```
lc_result = lc_openai.ChatCompletion.create(
    messages=messages,
    model="gpt-3.5-turbo",
    temperature=0
)
```

```
lc_result["choices"][0]['message']
```

```
    {'role': 'assistant', 'content': 'Hello! How can I assist you
today?'}
```

Swapping out model providers

```
lc_result = lc_openai.ChatCompletion.create(
    messages=messages,
    model="claude-2",
    temperature=0,
    provider="ChatAnthropic"
)
```

```
lc_result["choices"][0]['message']
```

```
    {'role': 'assistant', 'content': ' Hello!'}
```

ChatCompletion.stream

Original OpenAI call

```
for c in openai.ChatCompletion.create(
    messages = messages,
    model="gpt-3.5-turbo",
    temperature=0,
    stream=True
):
    print(c["choices"][0]['delta'].to_dict_recursive())
```

```
    {'role': 'assistant', 'content': ''}
    {'content': 'Hello'}
    {'content': '!'}
    {'content': ' How'}
    {'content': ' can'}
    {'content': ' I'}
    {'content': ' assist'}
    {'content': ' you'}
    {'content': ' today'}
    {'content': '?'}
    {}
```

LangChain OpenAI wrapper call

```
for c in lc_openai.ChatCompletion.create(
    messages = messages,
    model="gpt-3.5-turbo",
    temperature=0,
    stream=True
):
    print(c["choices"][0]['delta'])
```

```
    {'role': 'assistant', 'content': ''}
    {'content': 'Hello'}
    {'content': '!'}
    {'content': ' How'}
    {'content': ' can'}
    {'content': ' I'}
    {'content': ' assist'}
    {'content': ' you'}
    {'content': ' today'}
    {'content': '?'}
    {}
```

Swapping out model providers

```
for c in lc_openai.ChatCompletion.create(
    messages = messages,
    model="claude-2",
    temperature=0,
    stream=True,
    provider="ChatAnthropic",
):
    print(c["choices"][0]['delta'])
```

```
    {'role': 'assistant', 'content': ' Hello'}
    {'content': '!'}
    {}
```