Structured Output Parser

While the Pydantic/JSON parser is more powerful, we initially experimented data structures having text fields only.

```
from langchain.output_parsers import StructuredOutputParser, ResponseSchema
from langchain.prompts import PromptTemplate, ChatPromptTemplate,
HumanMessagePromptTemplate
from langchain.llms import OpenAI
from langchain.chat_models import ChatOpenAI
```

Here we define the response schema we want to receive.

```
response_schemas = [
    ResponseSchema(name="answer", description="answer to the user's question"),
    ResponseSchema(name="source", description="source used to answer the user's
question, should be a website.")
]
output_parser = StructuredOutputParser.from_response_schemas(response_schemas)
```

We now get a string that contains instructions for how the response should be formatted, and we then insert that into our prompt.

```
format_instructions = output_parser.get_format_instructions()
prompt = PromptTemplate(
    template="answer the users question as best as
possible.\n{format_instructions}\n{question}",
    input_variables=["question"],
    partial_variables={"format_instructions": format_instructions}
)
```

We can now use this to format a prompt to send to the language model, and then parse the returned result.

```
model = OpenAI(temperature=0)
```

Skip to main content

```
output = model(_input.to_string())
```

```
output_parser.parse(output)
```

```
{'answer': 'Paris', 'source': 'https://en.wikipedia.org/wiki/Paris'}
```

And here's an example of using this in a chat model

```
chat_model = ChatOpenAI(temperature=0)
```

```
prompt = ChatPromptTemplate(
    messages=[
        HumanMessagePromptTemplate.from_template("answer the users question as
best as possible.\n{format_instructions}\n{question}")
    ],
    input_variables=["question"],
    partial_variables={"format_instructions": format_instructions}
)
```

```
_input = prompt.format_prompt(question="what's the capital of france")
output = chat_model(_input.to_messages())
```

```
output_parser.parse(output.content)
```

```
{'answer': 'Paris', 'source': 'https://en.wikipedia.org/wiki/Paris'}
```