Serialization

This notebook covers how to serialize chains to and from disk. The serialization format we use is json or yaml. Currently, only some chains support this type of serialization. We will grow the number of supported chains over time.

Saving a chain to disk

First, let's go over how to save a chain to disk. This can be done with the `.save` method, and specifying a file path with a json or yaml extension.

```
from langchain import PromptTemplate, OpenAI, LLMChain

template = """Question: {question}

Answer: Let's think step by step."""
prompt = PromptTemplate(template=template, input_variables=
["question"])
llm_chain = LLMChain(prompt=prompt, llm=OpenAI(temperature=0),
verbose=True)
```

```
llm_chain.save("llm_chain.json")
```

Let's now take a look at what's inside this saved file

```
cat llm_chain.json
```

```
    {
        "memory": null,
        "verbose": true,
        "prompt": {
            "input_variables": [
                "question"
            ],
            "output_parser": null,
            "template": "Question: {question}\n\nAnswer: Let's think
```

```
step by step.",
            "template_format": "f-string"
        },
        "llm": {
            "model_name": "text-davinci-003",
            "temperature": 0.0,
            "max_tokens": 256,
            "top_p": 1,
            "frequency_penalty": 0,
            "presence_penalty": 0,
            "n": 1,
            "best_of": 1,
            "request_timeout": null,
            "logit_bias": {},
            "_type": "openai"
        },
        "output_key": "text",
        "_type": "llm_chain"
    }
```

Loading a chain from disk

We can load a chain from disk by using the `load_chain` method.

```
from langchain.chains import load_chain
```

API Reference:

- load_chain from `langchain.chains`

```
chain = load_chain("llm_chain.json")
```

```
chain.run("whats 2 + 2")
```

```
    > Entering new LLMChain chain...
    Prompt after formatting:
    Question: whats 2 + 2
```

```
    Answer: Let's think step by step.

    > Finished chain.
```

```
    ' 2 + 2 = 4'
```

Saving components separately

In the above example, we can see that the prompt and llm configuration information is saved in the same json as the overall chain. Alternatively, we can split them up and save them separately. This is often useful to make the saved components more modular. In order to do this, we just need to specify `llm_path` instead of the `llm` component, and `prompt_path` instead of the `prompt` component.

```
llm_chain.prompt.save("prompt.json")
```

```
cat prompt.json
```

```
    {
        "input_variables": [
            "question"
        ],
        "output_parser": null,
        "template": "Question: {question}\n\nAnswer: Let's think step
by step.",
        "template_format": "f-string"
    }
```

```
llm_chain.llm.save("llm.json")
```

```
cat llm.json
```

```
    {
        "model_name": "text-davinci-003",
        "temperature": 0.0,
        "max_tokens": 256,
        "top_p": 1,
        "frequency_penalty": 0,
        "presence_penalty": 0,
        "n": 1,
        "best_of": 1,
        "request_timeout": null,
        "logit_bias": {},
        "_type": "openai"
    }
```

```
config = {
    "memory": None,
    "verbose": True,
    "prompt_path": "prompt.json",
    "llm_path": "llm.json",
    "output_key": "text",
    "_type": "llm_chain",
}
import json

with open("llm_chain_separate.json", "w") as f:
    json.dump(config, f, indent=2)
```

```
cat llm_chain_separate.json
```

```
    {
      "memory": null,
      "verbose": true,
      "prompt_path": "prompt.json",
      "llm_path": "llm.json",
      "output_key": "text",
      "_type": "llm_chain"
    }
```

We can then load it in the same way

```
chain = load_chain("llm_chain_separate.json")
```

```
chain.run("whats 2 + 2")
```

```
> Entering new LLMChain chain...
Prompt after formatting:
Question: whats 2 + 2

Answer: Let's think step by step.

> Finished chain.
```

```
' 2 + 2 = 4'
```