Transformation

This notebook showcases using a generic transformation chain.

As an example, we will create a dummy transformation that takes in a super long text, filters the text to only the first 3 paragraphs, and then passes that into an LLMChain to summarize those.

```
from langchain.chains import TransformChain, LLMChain,
SimpleSequentialChain
from langchain.llms import OpenAI
from langchain.prompts import PromptTemplate
```

API Reference:

- TransformChain from `langchain.chains`
- LLMChain from `langchain.chains`
- SimpleSequentialChain from `langchain.chains`
- OpenAI from `langchain.llms`
- PromptTemplate from `langchain.prompts`

```
with open("../../state_of_the_union.txt") as f:
    state_of_the_union = f.read()
```

```
def transform_func(inputs: dict) -> dict:
    text = inputs["text"]
    shortened_text = "\n\n".join(text.split("\n\n")[:3])
    return {"output_text": shortened_text}


transform_chain = TransformChain(
    input_variables=["text"], output_variables=["output_text"],
transform=transform_func
)
```

```
template = """Summarize this text:

{output_text}
```

```
Summary:"""
prompt = PromptTemplate(input_variables=["output_text"],
template=template)
llm_chain = LLMChain(llm=OpenAI(), prompt=prompt)
```

```
sequential_chain = SimpleSequentialChain(chains=[transform_chain,
llm_chain])
```

```
sequential_chain.run(state_of_the_union)
```

```
    ' The speaker addresses the nation, noting that while last year
they were kept apart due to COVID-19, this year they are together
again. They are reminded that regardless of their political
affiliations, they are all Americans.'
```