Getting Started

Contents

In this tutorial, we will learn about:

- what a prompt template is, and why it is needed,
- how to create a prompt template,
- how to pass few shot examples to a prompt template,
- how to select examples for a prompt template.

What is a prompt template?

A prompt template refers to a reproducible way to generate a prompt. It contains a text string (“the template”), that can take in a set of parameters from the end user and generate a prompt.

The prompt template may contain:

- instructions to the language model,
- a set of few shot examples to help the language model generate a better response,
- a question to the language model.

The following code snippet contains an example of a prompt template:

```
from langchain import PromptTemplate
```

Skip to main content

```
template = """
I want you to act as a naming consultant for new companies.

Here are some examples of good company names:

- search engine, Google
- social media, Facebook
- video sharing, YouTube

The name should be short, catchy and easy to remember.

What is a good name for a company that makes {product}?
"""

prompt = PromptTemplate(
    input_variables=["product"],
    template=template,
)
```

Create a prompt template

You can create simple hardcoded prompts using the `PromptTemplate` class. Prompt templates can take any number of input variables, and can be formatted to generate a prompt.

```
from langchain import PromptTemplate

# An example prompt with no input variables
no_input_prompt = PromptTemplate(input_variables=[], template="Tell me a joke.")
no_input_prompt.format()
# -> "Tell me a joke."

# An example prompt with one input variable
one_input_prompt = PromptTemplate(input_variables=["adjective"], template="Tell me
a {adjective} joke.")
one_input_prompt.format(adjective="funny")
# -> "Tell me a funny joke."

# An example prompt with multiple input variables
multiple_input_prompt = PromptTemplate(
    input_variables=["adjective", "content"],
    template="Tell me a {adjective} joke about {content}."
)
multiple_input_prompt.format(adjective="funny", content="chickens")
# -> "Tell me a funny joke about chickens."
```

You can create custom prompt templates that format the prompt in any way you want. For

Skip to main content

> **Note**
>
> Currently, the template should be formatted as a Python f-string. We also support Jinja2 templates (see Using Jinja templates). In the future, we will support more templating languages such as Mako.

Load a prompt template from LangChainHub

LangChainHub contains a collection of prompts which can be loaded directly via LangChain.

```
from langchain.prompts import load_prompt

prompt = load_prompt("lc://prompts/conversation/prompt.json")
prompt.format(history="", input="What is 1 + 1?")
```

You can read more about LangChainHub and the prompts available with it here.

Pass few shot examples to a prompt template

Few shot examples are a set of examples that can be used to help the language model generate a better response.

To generate a prompt with few shot examples, you can use the `FewShotPromptTemplate`. This class takes in a `PromptTemplate` and a list of few shot examples. It then formats the prompt template with the few shot examples.

In this example, we'll create a prompt to generate word antonyms.

```
from langchain import PromptTemplate, FewShotPromptTemplate


# First, create the list of few shot examples.
examples = [
    {"word": "happy", "antonym": "sad"},
    {"word": "tall", "antonym": "short"},
]

# Next, we specify the template to format the examples we have provided.
# We use the `PromptTemplate` class for this
```

Skip to main content

```
Word: {word}
Antonym: {antonym}\n
"""
example_prompt = PromptTemplate(
    input_variables=["word", "antonym"],
    template=example_formatter_template,
)

# Finally, we create the `FewShotPromptTemplate` object.
few_shot_prompt = FewShotPromptTemplate(
    # These are the examples we want to insert into the prompt.
    examples=examples,
    # This is how we want to format the examples when we insert them into the
prompt.
    example_prompt=example_prompt,
    # The prefix is some text that goes before the examples in the prompt.
    # Usually, this consists of intructions.
    prefix="Give the antonym of every input",
    # The suffix is some text that goes after the examples in the prompt.
    # Usually, this is where the user input will go
    suffix="Word: {input}\nAntonym:",
    # The input variables are the variables that the overall prompt expects.
    input_variables=["input"],
    # The example_separator is the string we will use to join the prefix,
examples, and suffix together with.
    example_separator="\n\n",
)

# We can now generate a prompt using the `format` method.
print(few_shot_prompt.format(input="big"))
# -> Give the antonym of every input
# ->
# -> Word: happy
# -> Antonym: sad
# ->
# -> Word: tall
# -> Antonym: short
# ->
# -> Word: big
# -> Antonym:
```

Select examples for a prompt template

If you have a large number of examples, you can use the `ExampleSelector` to select a subset of examples that will be most informative for the Language Model. This will help you generate a prompt that is more likely to generate a good response.

Below, we'll use the `LengthBasedExampleSelector`, which selects examples based on the length

Skip to main content

the length of the context window. For longer inputs, it will select fewer examples to include, while for shorter inputs it will select more.

We'll continue with the example from the previous section, but this time we'll use the `LengthBasedExampleSelector` to select the examples.

```
from langchain.prompts.example_selector import LengthBasedExampleSelector


# These are a lot of examples of a pretend task of creating antonyms.
examples = [
    {"word": "happy", "antonym": "sad"},
    {"word": "tall", "antonym": "short"},
    {"word": "energetic", "antonym": "lethargic"},
    {"word": "sunny", "antonym": "gloomy"},
    {"word": "windy", "antonym": "calm"},
]

# We'll use the `LengthBasedExampleSelector` to select the examples.
example_selector = LengthBasedExampleSelector(
    # These are the examples is has available to choose from.
    examples=examples,
    # This is the PromptTemplate being used to format the examples.
    example_prompt=example_prompt,
    # This is the maximum length that the formatted examples should be.
    # Length is measured by the get_text_length function below.
    max_length=25,
)

# We can now use the `example_selector` to create a `FewShotPromptTemplate`.
dynamic_prompt = FewShotPromptTemplate(
    # We provide an ExampleSelector instead of examples.
    example_selector=example_selector,
    example_prompt=example_prompt,
    prefix="Give the antonym of every input",
    suffix="Word: {input}\nAntonym:",
    input_variables=["input"],
    example_separator="\n\n",
)

# We can now generate a prompt using the `format` method.
print(dynamic_prompt.format(input="big"))
# -> Give the antonym of every input
# ->
# -> Word: happy
# -> Antonym: sad
# ->
# -> Word: tall
# -> Antonym: short
# ->
# -> Word: energetic
```

Skip to main content

```
# ->
# -> Word: sunny
# -> Antonym: gloomy
# ->
# -> Word: windy
# -> Antonym: calm
# ->
# -> Word: big
# -> Antonym:
```

In contrast, if we provide a very long input, the `LengthBasedExampleSelector` will select fewer examples to include in the prompt.

```
long_string = "big and huge and massive and large and gigantic and tall and much
much much much much bigger than everything else"
print(dynamic_prompt.format(input=long_string))
# -> Give the antonym of every input

# -> Word: happy
# -> Antonym: sad
# ->
# -> Word: big and huge and massive and large and gigantic and tall and much much
much much much bigger than everything else
# -> Antonym:
```

LangChain comes with a few example selectors that you can use. For more details on how to use them, see Example Selectors.

You can create custom example selectors that select examples based on any criteria you want. For more details on how to do this, see Creating a custom example selector.