

# HuggingFace Pipeline\_Free

## Goal

- Use a free model (like from Hugging Face)
  - Use LangChain features (prompt template → model → output parser)
  - Use a Hugging Face or LangChain API key (both free)
  - Not download models locally
- 

## What You're Doing With OpenAI

You're doing:

1. Setting environment variables for OpenAI and Langsmith
  2. Creating a **ChatOpenAI** model
  3. Passing messages using **ChatPromptTemplate**
  4. Building a **chain** of prompt → model → output parser
  5. Getting string output
- 

## Can this be done with a free HF model?

YES — with changes. Here's the equivalent setup using Hugging Face Inference API (no downloads).

---

## Step-by-Step Replacement

### 1. Requirements

```
pip install langchain langchain-core langchain-huggingface huggingface_hub
```

## 2. HuggingFace API Token (free)

Get it from:

<https://huggingface.co/settings/tokens>

Create one with **"read" access**. It's free.

```
import os
from dotenv import load_dotenv
load_dotenv()

os.environ["HUGGINGFACEHUB_API_TOKEN"] = os.getenv("HUGGINGFACEHUB_API_TOKEN")
```

## 3. Use a Hugging Face Chat Model

Use a **chat-capable model** (e.g., [HuggingFaceH4/zephyr-7b-beta](#) )

```
from langchain_huggingface import HuggingFaceEndpoint

llm = HuggingFaceEndpoint(
    repo_id="HuggingFaceH4/zephyr-7b-beta",
    task="text-generation", # this is usually the correct task for chat-type models
    temperature=0.7,
    max_new_tokens=512
)
```

```

Generative AI refers to the use of machine learning algorithms to create original and novel content, rather than just process or analyze existing d
How does generative AI work?

Generative AI uses a process called "generative modeling" to create new content. This involves training a neural network on a large dataset of exis
One popular type of generative AI is called a "generative adversarial network" (GAN). This involves two neural networks working together: a generat
Benefits and limitations of generative AI:

Benefits:

1. Increased efficiency: Generative AI can produce large amounts of content quickly and accurately, which can be useful in fields such as marketing
2. Enhanced creativity: By generating new and original content, generative AI can help to spark new ideas and perspectives, and can be a valuable t
3. Improved accuracy: Generative AI can help to identify patterns and insights in data that may be difficult or time-consuming for humans to identi

Limitations:

1. Limited creativity: While generative AI can be very good at following patterns and generating new content based on existing data, it may not be
2. Accuracy limitations: Generative AI may not always be able to accurately replicate the nuances and subtleties of human language, speech, or othe
3. Data privacy concerns: Generative AI often requires large amounts of data to train its

```

```

result = llm.invoke("What is generative AI?")
print(result)

```

## 4. Prompt Template + Chain + OutputParser

```

from langchain_core.prompts import ChatPromptTemplate

prompt = ChatPromptTemplate.from_messages(
    [
        ("system", "You are an expert AI Engineer. Provide me answers based on
the questions"),
        ("user", "{input}")
    ]
)
print(prompt)

```

Output:

```
input_variables=['input'] input_types={} partial_variables={} messages=[SystemMessagePromptTemplate(prompt=PromptTemplate(input_variables=[], input_types={}, partial_variables={}, template='You are an expert AI Engineer. Provide me answers based on the questions'), additional_kwargs={}), HumanMessagePromptTemplate(prompt=PromptTemplate(input_variables=['input'], input_types={}, partial_variables={}, template='{input}'), additional_kwargs={})]
```

```
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser

# Prompt Template
prompt = ChatPromptTemplate.from_messages([
    ("system", "You are an expert AI engineer. Provide me answers based on the questions."),
    ("user", "{input}")
])

# Output parser
parser = StrOutputParser()

# Chain: prompt → model → output parser
chain = prompt | llm | parser

# Invoke
response = chain.invoke({"input": "Can you tell me about Langsmith?"})
print(response)
```

Output:


AI: Langsmith is a type of artificial intelligence (AI) that can understand and interpret human language. It is named after the linguist Noam Chomsky's theory of language acquisition, which suggests that humans are born with an innate ability to learn and understand language, called the language instinct or language faculty. Langsmith AI systems aim to mimic this natural language processing ability by learning the rules and structures of human language through large amounts of text data and neural networks. Langsmith AI can be used in a variety of applications, such as virtual assistants, chatbots, and language translation services. However, achieving full langsmith capabilities, such as understanding the nuances of human language and sarcasm, is still a major challenge in the field of AI research.

## ✓ Recap: What You Did (Without OpenAI)

Component	OpenAI Version	Free HF Version
Model	ChatOpenAI	HuggingFaceEndpoint (via API)
API Key	OPENAI_API_KEY	HUGGINGFACEHUB_API_TOKEN
Tracing (LangSmith)	Optional (same for both)	Same
Prompt Template	ChatPromptTemplate	Same
Output Parser	StrOutputParser	Same
Chain	`prompt`	llm

## ⚠ Important Notes

Concern	Answer
✓ Free to Use?	Yes, up to <b>30k tokens/day</b> for most accounts (HF API)
✗ Download model?	No, uses <b>Hugging Face Inference API</b>
✗ Do I need GPU locally?	No, all computation is done on Hugging Face cloud
? Can I use any model?	Use models with <b>Inference API enabled</b> (shown with rocket icon 🚀)

Concern	Answer
 Replaceable with other models?	Yes! E.g. <code>deepseek-ai/deepseek-coder</code> , <code>mistralai/Mistral-7B</code>