In [ ]:

In [ ]:
```
# ChatOpenAI_ChatGroq_project
```

In [152…
```
# load environment variables from the ".env" file.

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

Out[152… True

ChatGroq

In [ ]:

In [156…
```
# import "GROQ API key"

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

In [158…
```
# Import the ChatGroq

from langchain_groq import ChatGroq
```

In [206…
```
# Initialize the ChatGroq LLM using "llama3-8b-8192"

model = ChatGroq(model="llama3-8b-8192")
```

In [208…
```
#  import SystemMessage, and HumanMessage in a chat prompt.

from langchain_core.messages import SystemMessage, HumanMessage
```

In [210…
```
# Create a prompt template with a system message and a HumanMessage.

SystemMessage_HumanMessage = [
    SystemMessage(content="Please convert the human message from English to Nepali."),
    HumanMessage(content="Where are you from")
]
```

In [212…
```
# send the message to the model="llama3-8b-8192", and get its response

output = model.invoke(SystemMessage_HumanMessage)
```

In [214…
```
print(output)
```

```
content="Here's the translation:\n\nकहाँ हरुका हुनुहुन्छ?\n\n(kahaa haruka hunuhunchha?)\n\n(Note: The translation is in Devanagari script, which is the official script of Nepal. If you want the Romanization of the Nepali text, I can provide t
hat as well!)" additional_kwargs={} response_metadata={'token_usage': {'completion_tokens': 67, 'prompt_tokens': 30, 'total_tokens': 97, 'completion_time': 0.061323213, 'prompt_time': 0.007129426, 'queue_time': 0.024947986, 'total_tim
e': 0.068452639}, 'model_name': 'llama3-8b-8192', 'system_fingerprint': 'fp_5b339000ab', 'finish_reason': 'stop', 'logprobs': None} id='run--a27eca63-7b41-4303-a9c1-3bf16043a517-0' usage_metadata={'input_tokens': 30, 'output_tokens': 6
7, 'total_tokens': 97}
```

In [216…
```
# See the model's output
print(output.content)
```

```
Here's the translation:

कहाँ हरुका हुनुहुन्छ?

(kahaa haruka hunuhunchha?)

(Note: The translation is in Devanagari script, which is the official script of Nepal. If you want the Romanization of the Nepali text, I can provide that as well!)
```

In [224…
```
# Import StrOutputParser

from langchain_core.output_parsers import StrOutputParser
```

In [226…
```
# Make a parser to get plain text from the LLM output.

stroutput_parser =StrOutputParser()
```

In [228…
```
# see the plain text
stroutput_parser.invoke(output)
```

Out[228… "Here's the translation:\n\nकहाँ हरुका हुनुहुन्छ?\n\n(kahaa haruka hunuhunchha?)\n\n(Note: The translation is in Devanagari script, which is the official script of Nepal. If you want the Romanization of the Nepali text, I can provide that as well!)"

In [230…
```
# Use LangChain Expression Language (LCEL) to make a pipeline.
# use model and parser to make a pipeline

pipeline_chain=model|stroutput_parser
```

In [232…
```
# Get the final result with passing the human message through the model and then the parser.

pipeline_chain.invoke(SystemMessage_HumanMessage)
```

Out[232… 'In Nepali:\n\nतिमी कहाँबाट हुनुहुन्छ? (timi kahaanbāt hunuhunchha?)\n\nBreakdown:\n\n* तिमी (timi) means "you"\n* कहाँ (kahaan) means "where"\n* बाट (bāt) is a preposition meaning "from"\n* हुनुहुन्छ (hunuhunchha) is a polite way of saying "are you f
rom"\n\nNote: In Nepali, the sentence structure is often more formal and polite than in English, so this translation may sound more formal than the original sentence.'

ChatOpenAI

In [286…
```
# import "OPENAI_API_KEY"

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

In [288…
```
# Load ChatOpenAI
from langchain_openai import ChatOpenAI
```

In [290…
```
#  import HumanMessage
from langchain_core.prompts import ChatPromptTemplate
```

In [292…
```
# Create a templete to change text into the certain language

ChatPrompt_Template = "Please translate the following text into {language}."
```

In [294…
```
# Create a prompt with a system message and a user message.

prompt_system_user = ChatPromptTemplate.from_messages([
    ("system", ChatPrompt_Template),
    ("user", "{text}")
])
```

In [296…
```
# the prompt template including the expectated language and input text.
output = prompt_system_user.invoke({"language": "Nepali", "text": "Good Morning"})
```

In [298…
```
print(output.to_messages())
```

```
[SystemMessage(content='Please translate the following text into Nepali.', additional_kwargs={}, response_metadata={}), HumanMessage(content='Good Morning', additional_kwargs={}, response_metadata={})]
```

In [300…
```
from langchain_core.output_parsers import StrOutputParser
```

In [302…
```
stroutput_parser = StrOutputParser()
```

In [304…
```
# Make a chain that  sequentially  connects the prompt, model, and output parser

prompt_model_parser_chain = prompt_system_user | model | stroutput_parser
```

In [306…
```
output = prompt_model_parser_chain.invoke({"language": "Nepali", "text": "Good Morning"})
```

In [307…
```
print(output)
```

```
सुबही शुभकामना (Subahī śubhakāmnā)

Note: In Nepali, "Good Morning" is translated as "सुबही शुभकामना" (Subahī śubhakāmnā), which literally means "Morning wishes".
```

In [ ]: