🔷 Hugging Face Basics: Models, Tokenizers, and Pipelines

Hugging Face's transformers library provides easy access to **pre-trained NLP models** for tasks like text classification, translation, summarization, and chatbot applications.

1️⃣ Setting Up Hugging Face Transformers

Install the transformers library (if not installed):

```
pip install transformers
```

Test the installation:

```
import transformers
print(transformers.__version__)  # Should print the installed version
```

2️⃣ Hugging Face Pipelines: Quick and Easy NLP

Hugging Face **pipelines** provide an easy way to use pre-trained models without manually loading them.

```
from transformers import pipeline

# Sentiment Analysis Pipeline
classifier = pipeline("sentiment-analysis")
result = classifier("Hugging Face Transformers are amazing!")
print(result)
```

✓ Output:

```
[{'label': 'POSITIVE', 'score': 0.9998}]
```

🚀 Explanation:

• Loads a **pre-trained sentiment model** (distilbert-base-uncased-finetuned-sst-2-english).

• Tokenizes input, feeds it into the model, and returns **sentiment + confidence score**.

Supported Hugging Face Pipelines

Task	Pipeline Name	Example Model
Sentiment Analysis	"sentiment-analysis"	distilbert-base-uncased-finetuned-sst-2-english
Named Entity Recognition (NER)	"ner"	dbmdz/bert-large-cased-finetuned-conll03-english
Question Answering	"question-answering"	deepset/roberta-base-squad2
Text Generation	"text-generation"	gpt2
Translation	"translation_en_to_fr"	t5-small
Summarization	"summarization"	facebook/bart-large-cnn

3️⃣ Using Pretrained Models and Tokenizers

🔷 Loading a Model & Tokenizer Manually

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

model_name = "distilbert-base-uncased-finetuned-sst-2-english"

# Load Tokenizer and Model
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)
```

✓ Explanation:

• AutoTokenizer: Splits text into tokens for model input.

• AutoModelForSequenceClassification: Loads a model fine-tuned for classification tasks.

🔷 Tokenizing Text

```
text = "Hugging Face makes AI easy to use!"
inputs = tokenizer(text, return_tensors="pt")  # Convert text to tensor
print(inputs)
```

✓ Output (Tokenized Representation):

```
{
  'input_ids': tensor([[7592, 2229, 2227, 2209, 4773, 2000, 2224, 999]]),
  'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1]])
}
```

🚀 Explanation:

• Converts words into **token IDs**.

• Adds **attention masks** to ignore padding tokens.

4️⃣ Running Inference with a Model

Once tokenized, pass input to the model:

```
import torch

# Get Model Output
with torch.no_grad():
    output = model(**inputs)

# Convert output logits to probabilities
probs = torch.nn.functional.softmax(output.logits, dim=-1)
print(probs)
```

✓ Output:

```
tensor([[0.0015, 0.9985]])
```

🚀 Explanation:

• Model produces **logits** (raw scores).

• **Softmax** converts logits into **probabilities**.

5️⃣ Named Entity Recognition (NER)

Use an **NER pipeline** to detect entities in a sentence:

```
ner_pipeline = pipeline("ner", model="dbmdz/bert-large-cased-finetuned-
conll03-english")
text = "Hugging Face is a company based in New York and Paris."
ner_results = ner_pipeline(text)

for entity in ner_results:
    print(f"{entity['word']} → {entity['entity']} (Score:
{entity['score']:.2f})")
```

✓ Output:

```
Hugging → B-ORG (Score: 0.99)
Face → I-ORG (Score: 0.99)
```

```
New → B-LOC (Score: 0.98)
York → I-LOC (Score: 0.98)
Paris → B-LOC (Score: 0.99)
```

6️⃣ Question Answering

Pass a **context** and **question** to an NLP model:

```
qa_pipeline = pipeline("question-answering")

context = "Hugging Face was founded in 2016 by Clement Delangue, Julien
Chaumond, and Thomas Wolf."
question = "Who founded Hugging Face?"

answer = qa_pipeline(question=question, context=context)
print(answer)
```

✓ Output:

```
{'score': 0.999, 'start': 25, 'end': 66, 'answer': 'Clement Delangue, Julien
Chaumond, and Thomas Wolf'}
```

🚀 Explanation:

• Model extracts relevant **span** from the **context**.

7️⃣ Text Generation with GPT-2

Generate text using **GPT-2**:

```
generator = pipeline("text-generation", model="gpt2")

text = generator("AI will transform the world because", max_length=30,
num_return_sequences=2)
print(text)
```

✓ Output:

```
AI will transform the world because it has the power to revolutionize
industries, improve efficiency, and...
```

🚀 Explanation:

• GPT-2 **predicts the next words** given an input phrase.

8️⃣ Summarization

Summarize long text with **BART or T5**:

```
summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

text = """Hugging Face provides easy access to pre-trained models for natural language processing (NLP).
It supports various tasks like sentiment analysis, text generation, translation, and question answering."""
summary = summarizer(text, max_length=50, min_length=10, do_sample=False)
print(summary)
```

✓ Output:

```
["Hugging Face provides pre-trained NLP models for sentiment analysis, text generation, and more."]
```

🚀 Explanation:

• Model extracts key **summary points** from long text.

🔷 Summary of Code Examples

Task	Code Example
Sentiment Analysis	pipeline("sentiment-analysis")
Tokenizing Text	tokenizer("Hello World!", return_tensors="pt")
NER (Named Entity Recognition)	pipeline("ner")
Question Answering	pipeline("question-answering")
Text Generation	pipeline("text-generation", model="gpt2")
Summarization	pipeline("summarization", model="facebook/bart-large-cnn")

🚀 Next Steps

• 🎯 **Fine-tune models** for custom datasets.

• 🎯 **Deploy models** using **Gradio** or **Hugging Face Spaces**.

• 🎯 **Optimize performance** for GPUs (A100) and M1-M3 Macs.