```python
# necessary libraries
from transformers import pipeline, AutoTokenizer,
AutoModelForSeq2SeqLM
from tqdm import tqdm
import os

# input and output folders
input_dir = "samples"
output_dir = "samples"
os.makedirs(input_dir, exist_ok=True)
os.makedirs(output_dir, exist_ok=True)

# Function to load summarization pipeline
def load_summarizer():
    model_name = "facebook/bart-large-cnn"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
    return pipeline("summarization", model=model, tokenizer=tokenizer)

# Function to chunk long text into smaller parts
def chunk_text(text, max_tokens=800):
    words = text.split()
    for i in range(0, len(words), max_tokens):
        yield " ".join(words[i:i + max_tokens])

# summarizing one document
def summarize(text, summarizer):
    chunks = list(chunk_text(text))
    summary = ""
    for chunk in tqdm(chunks, desc="Summarizing"):
        result = summarizer(chunk, max_length=200, min_length=100,
do_sample=False)[0]
        summary += result['summary_text'] + " "
    return summary.strip()

# loading model once
summarizer = load_summarizer()
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"efec796811a4402298cba138b0c250c3","version_major":2,"vers
ion_minor":0}
```

{"model_id":"830f8de1ae964bf180b761168cef9872","version_major":2,"version_minor":0}

{"model_id":"5f59e2ff98a34c1eb9dc2dce2ff3fb54","version_major":2,"version_minor":0}

{"model_id":"e8396f2fd6764f7dbe2d1e9374a584de","version_major":2,"version_minor":0}

{"model_id":"14bb2111c75549978c169d287e88ea4c","version_major":2,"version_minor":0}

{"model_id":"4681e4b45c3e441ab3ed0f326eb69f8d","version_major":2,"version_minor":0}

```
Device set to use cpu
```

```
# All input files Calling
input_files = sorted([f for f in os.listdir(input_dir) if
f.startswith("input") and f.endswith(".txt")])

# looping through files and generate summaries
for input_file in input_files:
    input_path = os.path.join(input_dir, input_file)
    number = input_file.replace("input", "").replace(".txt", "")
    output_file = f"summary{number}.txt"
    output_path = os.path.join(output_dir, output_file)

    with open(input_path, 'r', encoding='utf-8') as f:
        text = f.read()

    print(f" Processing {input_file}...")
    summary = summarize(text, summarizer)

    with open(output_path, 'w', encoding='utf-8') as f:
        f.write(summary)

    print(f" Saved summary to {output_file}\n")

print(" All summaries complete!")
```

```
 Processing input1.txt...

Summarizing:   0%|          | 0/1 [00:00<?, ?it/s]Your max_length is
set to 200, but your input_length is only 179. Since this is a
summarization task, where outputs shorter than the input are typically
wanted, you might consider decreasing max_length manually, e.g.
summarizer('...', max_length=89)
Summarizing: 100%|██████████| 1/1 [00:26<00:00, 26.97s/it]
```

```
 Saved summary to summary1.txt

 Processing input2.txt...

Summarizing:   0%|          | 0/1 [00:00<?, ?it/s]Your max_length is
set to 200, but your input_length is only 180. Since this is a
summarization task, where outputs shorter than the input are typically
wanted, you might consider decreasing max_length manually, e.g.
summarizer('...', max_length=90)
Summarizing: 100%|██████████| 1/1 [00:26<00:00, 26.44s/it]

 Saved summary to summary2.txt

 Processing input3.txt...

Summarizing:   0%|          | 0/1 [00:00<?, ?it/s]Your max_length is
set to 200, but your input_length is only 182. Since this is a
summarization task, where outputs shorter than the input are typically
wanted, you might consider decreasing max_length manually, e.g.
summarizer('...', max_length=91)
Summarizing: 100%|██████████| 1/1 [00:29<00:00, 29.45s/it]

 Saved summary to summary3.txt

 Processing input4.txt...

Summarizing:   0%|          | 0/1 [00:00<?, ?it/s]Your max_length is
set to 200, but your input_length is only 163. Since this is a
summarization task, where outputs shorter than the input are typically
wanted, you might consider decreasing max_length manually, e.g.
summarizer('...', max_length=81)
Summarizing: 100%|██████████| 1/1 [00:33<00:00, 33.94s/it]

 Saved summary to summary4.txt

 Processing input5.txt...

Summarizing:   0%|          | 0/1 [00:00<?, ?it/s]Your max_length is
set to 200, but your input_length is only 163. Since this is a
summarization task, where outputs shorter than the input are typically
wanted, you might consider decreasing max_length manually, e.g.
summarizer('...', max_length=81)
Summarizing: 100%|██████████| 1/1 [00:28<00:00, 28.22s/it]

 Saved summary to summary5.txt

 All summaries complete!
```