```
!pip install evaluate
!pip install rouge_score
```

显示隐藏的输出项

```
# ---------- 0. Config ----------
MAX_EXAMPLES = 1000 # change to None to run full test set (may be large)
DEVICE = 'cuda' if __import__('torch').cuda.is_available() else 'cpu'
SMOL_MODEL = 'HuggingFaceTB/SmolLM-135M-Instruct'
PEGASUS_MODEL = 'google/pegasus-cnn_dailymail'

MAX_TOKENS = 90
MIN_LENGTH = 30
```

```
# ---------- 1. Imports ----------
from transformers import AutoTokenizer, AutoModelForCausalLM, AutoModelForSeq2SeqLM
from datasets import load_dataset
import evaluate
import torch
from tqdm.auto import tqdm
import pandas as pd
import os,json
```

```
# ---------- 2. Helpers: load data ----------


def load_cnn_dailymail(split='test', max_examples=None):
    ds = load_dataset('cnn_dailymail', '3.0.0', split=split)
    if max_examples is not None:
        ds = ds.select(range(max_examples))
    return ds
```

```
# ---------- 3. Model loaders ----------
def load_smol(model_name=SMOL_MODEL):
    tok = AutoTokenizer.from_pretrained(model_name, use_fast=True, trust_remote_code=True,chat_template=None)
    tok.padding_side = 'left'  # Left padding to avoid decoder-only warnings
    model = AutoModelForCausalLM.from_pretrained(model_name,trust_remote_code=True).to(DEVICE)
# ensure pad token
    if tok.pad_token is None:
        tok.add_special_tokens({'pad_token': '[PAD]'})
        model.resize_token_embeddings(len(tok))
    return tok, model


def load_pegasus(model_name=PEGASUS_MODEL):
    tok = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name).to(DEVICE)
    return tok, model
```

```
# ---------- 4. Prompt definitions (Unified for 1-2 Sentence Summary) ----------

def truncate_article(text, tokenizer, max_tokens=512):
        """Truncate article to ~512 tokens to prevent overflow when few-shot examples are added."""
        tokens = tokenizer.encode(text, truncation=True, max_length=max_tokens)
        return tokenizer.decode(tokens, skip_special_tokens=True)

# instruction prompt
def prompt_instruction(article, tokenizer=None):
        if tokenizer:
                article = truncate_article(article, tokenizer)
        return (
                "You are an advanced news summarization assistant.\n"
                "Write a detailed, factual summary of the following article in 3-4 complete sentences.\n"
                "Focus on the key events, people, and outcomes. Avoid repetition or speculation.\n\n"
                f"Article:\n{article.strip()}\n\n"
                "Summary:"
        )

FEW_SHOT_EXAMPLES = [
        (
                # 示例：篇幅更长的新闻（≈10句）
                "A powerful winter storm has blanketed much of the northeastern United States in heavy snow, disrupting trav
                "New York City received more than eight inches of snow overnight, while parts of Massachusetts and Connectic
                "Thousands of flights were canceled or delayed, leaving travelers stranded at major airports. "
                "Commuters faced treacherous road conditions, and authorities urged residents to stay home unless absolutely n
                "In Boston, city officials declared a snow emergency and deployed hundreds of plows to clear main roads. "
                "Utility companies reported widespread power outages as strong winds knocked down trees and power lines. "
```

```
                "Amtrak suspended most regional train services until conditions improved. "
                "Meteorologists say the storm is expected to weaken by Thursday afternoon but warned of another system devel
                "Despite the disruption, some residents took advantage of the snowfall to go sledding and build snowmen in
                "Officials reminded the public to check on elderly neighbors and keep emergency supplies at home in case of

                #  三到四句总结（对应输出）
                "Summary: A major winter storm brought heavy snow and high winds to the U.S. Northeast, disrupting travel a
                "More than a foot of snow fell in some areas, causing widespread flight cancellations and power outages. "
                "Officials declared emergencies, urged residents to stay indoors, and warned of another storm approaching next
                "The system is expected to weaken by Thursday afternoon as cleanup efforts continue."
        )
]


def prompt_few_shot(article, tokenizer=None):
        if tokenizer:
                article = truncate_article(article, tokenizer)
        #  用单个代表性示例
        example_article, example_summary = FEW_SHOT_EXAMPLES[0]
        return (
                "You are an advanced summarization assistant.\n"
                "Here are examples of how to summarize news articles in 3-4 sentences:\n\n"
                f"Example article:\n{example_article}\n{example_summary}\n\n"
                f"Now summarize the following article in 3-4 sentences:\n{article}\n\nSummary:"
        )
```

```
# ---------- 5. Generation wrappers ----------


# best-effort function to strip prompt prefix from generated token sequence
def strip_prefix(prompt, text):
    p = prompt.strip()
    t = text.strip()
    if not p:
        return t
    # if model echoes prompt, remove the longest common prefix
    if t.startswith(p):
        return t[len(p):].strip()
    # also try removing up to first newline after prompt
    return t

def generate_smol(tokenizer, model, prompts, strategy, max_new_tokens=90, min_length=50, batch_size=8):
        """
        Safe version for decoder-only models.
        Ensures one output per input prompt.
        Supports Beam, Top-p, Contrastive search, num_return_sequences>1.
        """
        model.eval()
        outputs = []

        for i in range(0, len(prompts), batch_size):
                batch = prompts[i:i+batch_size]
                inputs = tokenizer(batch, return_tensors='pt', padding=True, truncation=True, max_length=1024).to(DEVICE)

                with torch.no_grad():
                        gen = model.generate(
                                **inputs,
                                max_new_tokens=max_new_tokens,
                                min_length=min_length,
                                **strategy
                        )

                texts = tokenizer.batch_decode(gen, skip_special_tokens=True)

                #  Handle num_return_sequences > 1
                num_return = strategy.get('num_return_sequences', 1)
                if num_return > 1:
                    grouped = [texts[j*num_return:(j+1)*num_return] for j in range(len(batch))]
                    texts = [g[0] for g in grouped]      #  每个样本只保留第一个候选


                #  Ensure outputs match batch length
                assert len(texts) == len(batch), f"Mismatch in decoded outputs: {len(texts)} vs batch {len(batch)}"

                outputs.extend([strip_prefix(p, t) for p, t in zip(batch, texts)])

        #  Ensure total outputs match total prompts
        assert len(outputs) == len(prompts), f"Total outputs {len(outputs)} != total prompts {len(prompts)}"
        return outputs
```

```
def generate_pegasus(tokenizer, model, articles, max_new_tokens=90, min_length=50, strategy=None, batch_size=8):
    """
    Safe version for Seq2Seq models (PEGASUS).
    Ensures one output per article.
    """
    model.eval()
    strategy = strategy or {}
    outputs = []

    for i in range(0, len(articles), batch_size):
        batch = articles[i:i+batch_size]
        inputs = tokenizer(batch, return_tensors='pt', padding=True, truncation=True, max_length=1024).to(DEVICE)

        with torch.no_grad():
            gen = model.generate(
                **inputs,
                max_new_tokens=max_new_tokens,
                min_length=min_length,
                **strategy
            )

        texts = tokenizer.batch_decode(gen, skip_special_tokens=True)

        # Handle num_return_sequences > 1
        num_return = strategy.get('num_return_sequences', 1)
        if num_return > 1:
            texts = [texts[j * num_return] for j in range(len(batch))]

        # Ensure outputs match batch length
        assert len(texts) == len(batch), f"Mismatch in decoded outputs: {len(texts)} vs batch {len(batch)}"

        outputs.extend([t.strip() for t in texts])

    # Ensure total outputs match total articles
    assert len(outputs) == len(articles), f"Total outputs {len(outputs)} != total articles {len(articles)}"
    return outputs
```

```
# ---------- 6. Evaluation helpers ----------

rouge = evaluate.load('rouge')

def compute_rouge(preds, refs):
    # preds and refs: lists of strings
    # rouge.compute handles batch
    result = rouge.compute(predictions=preds, references=refs)
    # extract f-measure mid (if available)
    def _get(k):
        v = result.get(k)
        if hasattr(v, 'mid'):
            return v.mid.fmeasure
        return v
    return {'rouge1': _get('rouge1'), 'rouge2': _get('rouge2'), 'rougeL': _get('rougeL')}
```

```
/usr/local/lib/python3.12/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
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(
```

```
def run_experiments(
    max_examples=MAX_EXAMPLES,
    min_length=MIN_LENGTH,
    save_outputs_path='results.jsonl',
    mode='smol'    # 可选: 'smol' 或 'pegasus'
):
    """
    运行摘要实验（支持两种模式: SmolLM 或 PEGASUS）。
    不使用断点续跑。
    """
    # 加载数据
    ds = load_cnn_dailymail('test', max_examples=max_examples)
    articles = [ex['article'] for ex in ds]
    refs = [ex['highlights'] for ex in ds]

    experiments = []

    if mode == 'smol':
        print('Loading SmolLM...')
        tok_smol, model_smol = load_smol()
```

```
        all_experiments = [
                ("E1: Instruction+Greedy", lambda: [prompt_instruction(a, tok_smol) for a in articles], {'num_beams': 1,
                ("E2: Instruction+Beam", lambda: [prompt_instruction(a, tok_smol) for a in articles], {'num_beams': 2, 'd
                ("E3: FewShot+Beam", lambda: [prompt_few_shot(a, tok_smol) for a in articles], {'num_beams': 2, 'do_sampl
                ("E4: FewShot+TopP", lambda: [prompt_few_shot(a, tok_smol) for a in articles], {'do_sample': True, 'num_b
        ]

        for name, prompt_fn, strategy in all_experiments:
                print(f'🚀 Running {name}...')
                prompts = prompt_fn()
                preds = generate_smol(tok_smol, model_smol, prompts, strategy, max_new_tokens=MAX_TOKENS, min_length=MIN_LEN
                scores = compute_rouge(preds, refs)
                experiments.append({'name': name, 'scores': scores, 'preds': preds})
                torch.cuda.empty_cache()

    elif mode == 'pegasus':
        print('Loading PEGASUS...')
        tok_peg, model_peg = load_pegasus()
        strategy_peg = {'num_beams': 8, 'length_penalty': 0.8}
        preds_peg = generate_pegasus(tok_peg, model_peg, articles, max_new_tokens=MAX_TOKENS, min_length=MIN_LENGTH, strategy
        scores_peg = compute_rouge(preds_peg, refs)
        experiments.append({'name': 'PEGASUS', 'scores': scores_peg, 'preds': preds_peg})
        torch.cuda.empty_cache()

    else:
        raise ValueError("mode must be 'smol' or 'pegasus'")

    # 汇总结果
    rows = [{'method': exp['name'], **exp['scores']} for exp in experiments]
    df = pd.DataFrame(rows).set_index('method')

    # 保存 per-example JSONL
    with open(save_outputs_path, 'w', encoding='utf-8') as fh:
        for i in range(len(articles)):
            rec = {'id': ds[i]['id'], 'article': articles[i], 'reference': refs[i]}
            for exp in experiments:
                rec[exp['name']] = exp['preds'][i]
            fh.write(json.dumps(rec, ensure_ascii=False) + '\n')

    return experiments, df
```

```
# ---------- 8. If run as script ----------

if __name__ == '__main__':
    # --------- Step 1: 运行 SmolLM ----------
    torch.cuda.empty_cache()
    res_smol, df_smol = run_experiments(mode='smol', max_examples=MAX_EXAMPLES, min_length=MIN_LENGTH, save_outputs_path='smol_res
    print(df_smol)
    df_smol.to_csv('smol_rouge_scores.csv')

    # --------- Step 2: 释放 GPU，再运行 PEGASUS ----------
    torch.cuda.empty_cache()
    res_peg, df_peg = run_experiments(mode='pegasus', max_examples=MAX_EXAMPLES, min_length=MIN_LENGTH, save_outputs_path='pegasus
    print(df_peg)
    df_peg.to_csv('pegasus_rouge_scores.csv')

    # --------- Step 3: 拼接结果 ----------
    #df_smol = pd.read_csv('smol_rouge_scores.csv')
    df_total = pd.concat([df_smol, df_peg], ignore_index=False)
    print(df_total)
    df_total.to_csv('summary_rouge_scores.csv')

    print('\n✅ 已保存所有输出文件')

# End of notebook
```

```
Loading SmolLM...
🚀 Running E1: Instruction+Greedy...
🚀 Running E2: Instruction+Beam...
🚀 Running E3: FewShot+Beam...
🚀 Running E4: FewShot+TopP...
                         rouge1    rouge2    rougeL
method
E1: Instruction+Greedy  0.136200  0.016615  0.103861
E2: Instruction+Beam    0.137873  0.015332  0.103841
E3: FewShot+Beam        0.147683  0.017549  0.107364
E4: FewShot+TopP        0.148044  0.015964  0.106671
Loading PEGASUS...
model.safetensors: 100%                                  2.28G/2.28G [00:37<00:00, 92.3MB/s]
Some weights of PegasusForConditionalGeneration were not initialized from the model checkpoint at google/pegasus-cnn_dailymail and are newly
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
          rouge1  rouge2    rougeL
method
PEGASUS  0.321738  0.1335  0.237964
                         rouge1    rouge2    rougeL
method
E1: Instruction+Greedy  0.136200  0.016615  0.103861
E2: Instruction+Beam    0.137873  0.015332  0.103841
E3: FewShot+Beam        0.147683  0.017549  0.107364
E4: FewShot+TopP        0.148044  0.015964  0.106671
PEGASUS                 0.321738  0.133500  0.237964

✅ 已保存所有输出文件
```