assignment_3

October 24, 2025

[ ]:
```
# === IMPORTS AND CONFIG ===
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
```

[ ]:
```
# === CONFIGURATIONS ===
DEVICE = 'cuda' if __import__('torch').cuda.is_available() else 'cpu'
SMOL_MODEL = 'HuggingFaceTB/SmolLM-135M-Instruct'
PEGASUS_MODEL = 'google/pegasus-cnn_dailymail'

MAX_EXAMPLES = 1000 # Use a subset for faster evaluation
MAX_TOKENS = 90 # Max new tokens for summary
MIN_LENGTH = 30 # Min length for summary
```

[ ]:
```
# === DATA LOADING ===
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
```

[ ]:
```
# === 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
```

[ ]:
```
# === PROMPT DEFINITIONS ===

def truncate_article(text, tokenizer, max_tokens=512):
    """Truncates text to max_tokens to fit in context."""
    tokens = tokenizer.encode(text, truncation=True, max_length=max_tokens)
    return tokenizer.decode(tokens, skip_special_tokens=True)

def prompt_instruction(article, tokenizer=None):
    """Creates a zero-shot instruction prompt."""
    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:"
    )

# Define few-shot examples (using one representative example)
FEW_SHOT_EXAMPLES = [
    (
        "A powerful winter storm has blanketed much of the northeastern United␣
↪States in heavy snow, disrupting travel and closing schools across several␣
↪states. "
        "New York City received more than eight inches of snow overnight, while␣
↪parts of Massachusetts and Connecticut reported over a foot. "
        "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 necessary. "
        "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 developing early next week. "
        "Despite the disruption, some residents took advantage of the snowfall␣
↪to go sledding and build snowmen in city parks. "
        "Officials reminded the public to check on elderly neighbors and keep␣
↪emergency supplies at home in case of extended outages.",

        "Summary: A major winter storm brought heavy snow and high winds to the␣
↪U.S. Northeast, disrupting travel and shutting down schools. "
        "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 week. "
        "The system is expected to weaken by Thursday afternoon as cleanup␣
↪efforts continue."
    )
]


def prompt_few_shot(article, tokenizer=None):
    """Creates a one-shot prompt."""
    if tokenizer:
        article = truncate_article(article, tokenizer)

    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:"
    )
```

[ ]:
```
# === GENERATION WRAPPERS ===
def strip_prefix(prompt, text):
  """Removes the prompt from the start of the generated text."""
  p = prompt.strip()
  t = text.strip()
  if not p:
    return t
  if t.startswith(p):
    return t[len(p):].strip()
  return t
```

```
def generate_smol(tokenizer, model, prompts, strategy, max_new_tokens=90,␣
 ↪min_length=50, batch_size=8):
    """Handles generation for the decoder-only SmolLM."""
    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 cases where 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]
    return outputs

def generate_pegasus(tokenizer, model, articles, max_new_tokens=90,␣
 ↪min_length=50, strategy=None, batch_size=8):
    """Handles generation for the Seq2Seq PEGASUS model."""
    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))]

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

[ ]:
```
# === EVALUATION HELPER ===
rouge = evaluate.load('rouge')

def compute_rouge(preds, refs):
  """Computes ROUGE scores."""
  result = rouge.compute(predictions=preds, references=refs)
  # Helper to extract f-measure
  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')}
```

[ ]:
```
# === EXPERIMENT RUNNER ===
def run_experiments(
    max_examples=MAX_EXAMPLES,
    min_length=MIN_LENGTH,
    save_outputs_path='results.jsonl',
    mode='smol'
):
    """Runs all experiments for a given mode ('smol' or 'pegasus')."""
    # Load data
    ds = load_cnn_dailymail('test', max_examples=max_examples)
    articles = [ex['article'] for ex in ds]
    refs = [ex['highlights'] for ex in ds]

    experiments = []

    # --- Task 2: SmolLM Experiments ---
    if mode == 'smol':
        print('Loading SmolLM...')
        tok_smol, model_smol = load_smol()
        # Define the 4 strategies
        all_experiments = [
            ("E1: Instruction+Greedy", lambda: [prompt_instruction(a, tok_smol)␣
↪for a in articles], {'num_beams': 1, 'do_sample': False,␣
↪'no_repeat_ngram_size': 3}),
            ("E2: Instruction+Beam", lambda: [prompt_instruction(a, tok_smol)␣
↪for a in articles], {'num_beams': 2, 'do_sample': False, 'early_stopping':␣
↪True, 'no_repeat_ngram_size': 3}),
            ("E3: FewShot+Beam", lambda: [prompt_few_shot(a, tok_smol) for a in␣
↪articles], {'num_beams': 2, 'do_sample': False, 'early_stopping': True,␣
↪'no_repeat_ngram_size': 3}),
            ("E4: FewShot+TopP", lambda: [prompt_few_shot(a, tok_smol) for a in␣
↪articles], {'do_sample': True, 'num_beams': 1, 'top_p': 0.95, 'temperature':␣
↪0.7, 'no_repeat_ngram_size': 3}),
        ]

        # Run each SmolLM experiment
        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_LENGTH)
            scores = compute_rouge(preds, refs)
            experiments.append({'name': name, 'scores': scores, 'preds': preds})
            torch.cuda.empty_cache()

    # --- Task 3: PEGASUS Comparison ---
    elif mode == 'pegasus':
        print('Loading PEGASUS...')
        tok_peg, model_peg = load_pegasus()
        # Define PEGASUS strategy
        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=strategy_peg)
        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'")

    # --- RESULTS & SAVING ---
    # Collate results into a DataFrame
    rows = [{'method': exp['name'], **exp['scores']} for exp in experiments]
    df = pd.DataFrame(rows).set_index('method')

    # Save raw outputs
    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
```

```
[ ]: # === MAIN EXECUTION ===
if __name__ == '__main__':
    # Step 1: Run SmolLM experiments
    torch.cuda.empty_cache()
    res_smol, df_smol = run_experiments(mode='smol', max_examples=MAX_EXAMPLES,␣
↪min_length=MIN_LENGTH, save_outputs_path='smol_results.jsonl')
    print(df_smol)
    df_smol.to_csv('smol_rouge_scores.csv')

    # Step 2: Run PEGASUS experiment
    torch.cuda.empty_cache()
    res_peg, df_peg = run_experiments(mode='pegasus',␣
↪max_examples=MAX_EXAMPLES, min_length=MIN_LENGTH,␣
↪save_outputs_path='pegasus_results.jsonl')
    print(df_peg)
    df_peg.to_csv('pegasus_rouge_scores.csv')

    # Step 3: Combine and save all results
    #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')
```

```
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:   0%|          | 0.00/2.28G [00:00<?, ?B/s]

Some weights of PegasusForConditionalGeneration were not initialized from the
model checkpoint at google/pegasus-cnn_dailymail and are newly initialized:
['model.decoder.embed_positions.weight', 'model.encoder.embed_positions.weight']
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
```