```
!pip install evaluate
```

```
Requirement already satisfied: evaluate in
/usr/local/lib/python3.11/dist-packages (0.4.3)
Requirement already satisfied: datasets>=2.0.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (3.6.0)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (1.26.4)
Requirement already satisfied: dill in /usr/local/lib/python3.11/dist-
packages (from evaluate) (0.3.8)
Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (from evaluate) (2.2.3)
Requirement already satisfied: requests>=2.19.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (4.67.1)
Requirement already satisfied: xxhash in
/usr/local/lib/python3.11/dist-packages (from evaluate) (3.5.0)
Requirement already satisfied: multiprocess in
/usr/local/lib/python3.11/dist-packages (from evaluate) (0.70.16)
Requirement already satisfied: fsspec>=2021.05.0 in
/usr/local/lib/python3.11/dist-packages (from fsspec[http]>=2021.05.0-
>evaluate) (2025.3.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (0.31.1)
Requirement already satisfied: packaging in
/usr/local/lib/python3.11/dist-packages (from evaluate) (25.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0-
>evaluate) (3.18.0)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0-
>evaluate) (19.0.1)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0-
>evaluate) (6.0.2)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in
/usr/local/lib/python3.11/dist-packages (from fsspec[http]>=2021.05.0-
>evaluate) (3.11.18)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.7.0-
>evaluate) (4.13.2)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.7.0-
>evaluate) (1.1.0)
Requirement already satisfied: mkl_fft in
/usr/local/lib/python3.11/dist-packages (from numpy>=1.17->evaluate)
(1.3.8)
Requirement already satisfied: mkl_random in
/usr/local/lib/python3.11/dist-packages (from numpy>=1.17->evaluate)
```

```
(1.2.4)
Requirement already satisfied: mkl_umath in
/usr/local/lib/python3.11/dist-packages (from numpy>=1.17->evaluate)
(0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.17->evaluate) (2025.1.0)
Requirement already satisfied: tbb4py in
/usr/local/lib/python3.11/dist-packages (from numpy>=1.17->evaluate)
(2022.1.0)
Requirement already satisfied: mkl-service in
/usr/local/lib/python3.11/dist-packages (from numpy>=1.17->evaluate)
(2.4.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0-
>evaluate) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0-
>evaluate) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0-
>evaluate) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0-
>evaluate) (2025.4.26)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->evaluate)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas->evaluate)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas->evaluate)
(2025.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (1.6.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (6.4.3)
Requirement already satisfied: propcache>=0.2.0 in
```

```
/usr/local/lib/python3.11/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (1.20.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas->evaluate) (1.17.0)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy>=1.17-
>evaluate) (2024.2.0)
Requirement already satisfied: tbb==2022.* in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy>=1.17-
>evaluate) (2022.1.0)
Requirement already satisfied: tcmlib==1.* in
/usr/local/lib/python3.11/dist-packages (from tbb==2022.*->mkl-
>numpy>=1.17->evaluate) (1.3.0)
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from mkl_umath->numpy>=1.17-
>evaluate) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-
openmp<2026,>=2024->mkl->numpy>=1.17->evaluate) (2024.2.0)
```

```
!pip install jiwer
```

```
Requirement already satisfied: jiwer in
/usr/local/lib/python3.11/dist-packages (3.1.0)
Requirement already satisfied: click>=8.1.8 in
/usr/local/lib/python3.11/dist-packages (from jiwer) (8.1.8)
Requirement already satisfied: rapidfuzz>=3.9.7 in
/usr/local/lib/python3.11/dist-packages (from jiwer) (3.13.0)
```

```
import os
import random
import numpy as np
import pandas as pd
from tqdm.auto import tqdm
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import (
    T5ForConditionalGeneration,
    T5Tokenizer,
    Seq2SeqTrainingArguments,
    Seq2SeqTrainer,
    DataCollatorForSeq2Seq,
    EarlyStoppingCallback
)
from datasets import Dataset as HFDataset
import nltk
```

```
from nltk.metrics.distance import edit_distance
from sklearn.model_selection import train_test_split
import evaluate
import gc
import re

# Download necessary nltk resources
nltk.download('punkt')

# Set random seeds for reproducibility
random.seed(42)
np.random.seed(42)
torch.manual_seed(42)
torch.cuda.manual_seed_all(42)

# Define paths
DATA_DIR =
"/kaggle/input/transformer-mode-for-spelling-correction/wiki-split-
master"  # Default location for Kaggle dataset
CACHE_DIR = "/kaggle/working/cache"  # For saving model checkpoints
OUTPUT_DIR = "/kaggle/working/outputs"  # For saving outputs

# Create cache and output directories if they don't exist
os.makedirs(CACHE_DIR, exist_ok=True)
os.makedirs(OUTPUT_DIR, exist_ok=True)

# Check if GPU is available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
```

```
2025-05-14 00:58:53.945011: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to
register cuFFT factory: Attempting to register factory for plugin
cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
E0000 00:00:1747184333.967350    1893 cuda_dnn.cc:8310] Unable to
register cuDNN factory: Attempting to register factory for plugin
cuDNN when one has already been registered
E0000 00:00:1747184333.974331    1893 cuda_blas.cc:1418] Unable to
register cuBLAS factory: Attempting to register factory for plugin
cuBLAS when one has already been registered
```

```
Using device: cuda
```

```
[nltk_data] Downloading package punkt to /usr/share/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
# 1. Load the WikiSplit dataset
def load_data(file_path, max_samples=None):
    print(f"Loading data from {file_path}")
    df = pd.read_csv(file_path, sep='\t', header=None,
names=['correct', 'split'])
    # Keep only the first column (correct sentences)
    df = df[['correct']]
    if max_samples:
        df = df.sample(n=min(max_samples, len(df)), random_state=42)
    return df

# Set the maximum number of samples to use (for memory efficiency)
# Using just 10% of the original sizes for extreme memory efficiency
MAX_TRAIN_SAMPLES = 50000   # Reduced to 5k examples for training (10%
of original)
MAX_VAL_SAMPLES = 5000      # Reduced to 500 examples for validation
MAX_TEST_SAMPLES = 5000     # Reduced to 500 examples for testing

# Load the datasets (with limits for memory efficiency)
train_df = load_data(os.path.join(DATA_DIR, "tune.tsv"),
MAX_TRAIN_SAMPLES)
val_df = load_data(os.path.join(DATA_DIR, "validation.tsv"),
MAX_VAL_SAMPLES)
test_df = load_data(os.path.join(DATA_DIR, "test.tsv"),
MAX_TEST_SAMPLES)

print(f"Train samples: {len(train_df)}")
print(f"Validation samples: {len(val_df)}")
print(f"Test samples: {len(test_df)}")

# 2. Generate sentences with spelling mistakes
def generate_misspelled_sentence(sentence, error_prob=0.3,
char_error_prob=0.7):
    """
    Generate a misspelled sentence by introducing various types of
spelling errors:
    - Character swapping
    - Character deletion
    - Character insertion
    - Character replacement

    Args:
        sentence: Original sentence
        error_prob: Probability of introducing an error to a word
        char_error_prob: Probability of character-level error (vs
word-level)

    Returns:
        Misspelled sentence
    """
    words = sentence.split()
    misspelled_words = []
    for word in words:
        # Skip very short words and punctuation
        if len(word) <= 2 or not any(c.isalpha() for c in word):
            misspelled_words.append(word)
            continue

        # Decide whether to introduce an error to this word
        if random.random() < error_prob:
            # Character-level errors
            if random.random() < char_error_prob:
                chars = list(word)
                error_type = random.choice(['swap', 'delete',
'insert', 'replace'])

                if error_type == 'swap' and len(word) >= 2:
                    # Swap two adjacent characters
                    idx = random.randint(0, len(chars) - 2)
                    chars[idx], chars[idx + 1] = chars[idx + 1],
chars[idx]

                elif error_type == 'delete' and len(word) >= 3:
                    # Delete a character
                    idx = random.randint(0, len(chars) - 1)
                    chars.pop(idx)

                elif error_type == 'insert':
                    # Insert a random character
                    idx = random.randint(0, len(chars))
                    chars.insert(idx,
random.choice('abcdefghijklmnopqrstuvwxyz'))

                elif error_type == 'replace' and len(word) >= 1:
                    # Replace a character with another random
character
                    idx = random.randint(0, len(chars) - 1)
                    new_char =
random.choice('abcdefghijklmnopqrstuvwxyz')
                    while new_char == chars[idx]:
                        new_char =
random.choice('abcdefghijklmnopqrstuvwxyz')
                    chars[idx] = new_char

                misspelled_words.append(''.join(chars))

            # Word-level errors (split or merge words)
            else:
                error_type = random.choice(['split', 'merge'])

                if error_type == 'split' and len(word) >= 4:
                    # Split the word into two parts
                    split_idx = random.randint(1, len(word) - 1)
                    misspelled_words.append(word[:split_idx])
                    misspelled_words.append(word[split_idx:])

                elif error_type == 'merge' and len(misspelled_words) > 
0 and len(misspelled_words[-1]) + len(word) <= 15:
                    # Merge with previous word (with some constraints)
                    prev_word = misspelled_words.pop()
                    misspelled_words.append(prev_word + word)

                else:
                    # Default to adding the word as is if merge isn't 
possible
                    misspelled_words.append(word)
        else:
            misspelled_words.append(word)

    return ' '.join(misspelled_words)

# Create a function to generate a dataset with spelling mistakes
def create_spelling_dataset(df, num_variants=1):
    """
    Create a dataset with spelling mistakes

    Args:
        df: DataFrame with correct sentences
        num_variants: Number of misspelled variants to generate per 
sentence

    Returns:
        DataFrame with misspelled and correct sentences
    """
    misspelled_data = []

    for _, row in tqdm(df.iterrows(), total=len(df), desc="Generating 
misspelled sentences"):
        correct = row['correct']

        # Skip sentences that are too long (for memory efficiency)
        if len(correct.split()) > 30:
            continue

        for _ in range(num_variants):
            misspelled = generate_misspelled_sentence(correct)
            misspelled_data.append({
                'misspelled': misspelled,
                'correct': correct
            })
    return pd.DataFrame(misspelled_data)

# Generate datasets with spelling mistakes
print("Generating datasets with spelling mistakes...")
train_data = create_spelling_dataset(train_df)
val_data = create_spelling_dataset(val_df)
test_data = create_spelling_dataset(test_df)

print(f"Generated train samples: {len(train_data)}")
print(f"Generated validation samples: {len(val_data)}")
print(f"Generated test samples: {len(test_data)}")

# Let's check a few examples to confirm our spelling mistakes
generation
print("\nSample data (first 3 examples):")
for i in range(min(3, len(train_data))):
    print(f"Original  : {train_data.iloc[i]['correct']}")
    print(f"Misspelled: {train_data.iloc[i]['misspelled']}")
    print()
```

```
Loading data from /kaggle/input/transformer-mode-for-spelling-
correction/wiki-split-master/tune.tsv
Loading data from /kaggle/input/transformer-mode-for-spelling-
correction/wiki-split-master/validation.tsv
Loading data from /kaggle/input/transformer-mode-for-spelling-
correction/wiki-split-master/test.tsv
Train samples: 5000
Validation samples: 5000
Test samples: 5000
Generating datasets with spelling mistakes...
```

{"model_id":"4160a513479d47ee8580cb25b6ef9fc0","version_major":2,"vers
ion_minor":0}

{"model_id":"f6758492838941e5a8a8c6507e074e1e","version_major":2,"vers
ion_minor":0}

{"model_id":"b3faa50d265f4a538daa315826bc9d44","version_major":2,"vers
ion_minor":0}

```
Generated train samples: 1962
Generated validation samples: 1970
Generated test samples: 1876

Sample data (first 3 examples):
Original  : He lived here with his wife Maria Hansen , the daughter of
the Danish / German astronomer Peter Andreas Hansen .
Misspelled: He lived hre with his wife aMria Hansne , ohe daughter of
theDanish / German astronomer Petter Andrdas Hansen .
```

```
Original  : Many groups want the cross to be included , while other
organizations , notably American Atheists , disagree .
Misspelled: Many groups want the cross to be included , while hother
organizations , notably American Atheists , dtisagree .

Original  : By 1759 , the Dutch East India Company 's trade had fallen
substantially , and operations were moved to Bombay , with Suratte
playing a subordinate role .
Misspelled: By 1759 , the Dutch Easth India Company 's trade
hadifallen suistantially ,and operations were moved to Bombay , with
uSratte playcng a subordinate role .


# 3. Create datasets in HuggingFace format
def create_hf_dataset(df):
    return HFDataset.from_pandas(df)

train_dataset = create_hf_dataset(train_data)
val_dataset = create_hf_dataset(val_data)
test_dataset = create_hf_dataset(test_data)

# 4. Preprocess data for the model
# We'll use T5 for this task (smaller than BART, better for memory
constraints)
# T5-small is a good choice for Kaggle's T4x2 environment
MODEL_NAME = "t5-base"
tokenizer = T5Tokenizer.from_pretrained(MODEL_NAME,
cache_dir=CACHE_DIR)

# Set max length based on our dataset analysis (adjust if needed)
MAX_SOURCE_LENGTH = 64   # Reduced from 128
MAX_TARGET_LENGTH = 64   # Reduced from 128

def preprocess_function(examples):
    """Tokenize the texts and prepare them for the model"""
    # For T5, we need to add a prefix for the task
    inputs = ["correct spelling: " + ex for ex in
examples["misspelled"]]
    targets = examples["correct"]

    model_inputs = tokenizer(
        inputs,
        max_length=MAX_SOURCE_LENGTH,
        padding="max_length",
        truncation=True
    )

    # Setup the tokenizer for targets
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
```

```python
            targets,
            max_length=MAX_TARGET_LENGTH,
            padding="max_length",
            truncation=True
        )

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

# Apply preprocessing to all datasets
print("Tokenizing datasets...")
train_dataset = train_dataset.map(
    preprocess_function,
    batched=True,
    remove_columns=train_dataset.column_names,
    desc="Preprocessing train dataset",
)

val_dataset = val_dataset.map(
    preprocess_function,
    batched=True,
    remove_columns=val_dataset.column_names,
    desc="Preprocessing validation dataset",
)

test_dataset = test_dataset.map(
    preprocess_function,
    batched=True,
    remove_columns=test_dataset.column_names,
    desc="Preprocessing test dataset",
)
```

```
You are using the default legacy behaviour of the <class
'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is
expected, and simply means that the `legacy` (previous) behavior will
be used so nothing changes for you. If you want to use the new
behaviour, set `legacy=False`. This should only be set if you
understand what it means, and thoroughly read the reason why this was
added as explained in
https://github.com/huggingface/transformers/pull/24565
```

```
Tokenizing datasets...
```

```
{"model_id":"2544084218ac4a949d2adaf0f7df9c89","version_major":2,"vers
ion_minor":0}
```

```
/usr/local/lib/python3.11/dist-packages/transformers/
tokenization_utils_base.py:3980: UserWarning: `as_target_tokenizer` is
deprecated and will be removed in v5 of Transformers. You can tokenize
your labels by using the argument `text_target` of the regular
```

```
`__call__` method (either in the same call as your input texts if you
use the same keyword arguments, or in a separate call.
  warnings.warn(

{"model_id":"6cb082113e6243a39749f0c50ecd9901","version_major":2,"vers
ion_minor":0}

{"model_id":"4763d31ddaf94782acc3201825cea4d0","version_major":2,"vers
ion_minor":0}

# 5. Define Evaluation Metrics
wer_metric = evaluate.load("wer")
cer_metric = evaluate.load("cer")

def compute_metrics(eval_preds):
    preds, labels = eval_preds

    # Replace -100 in labels with the pad token id for decoding
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)

    # Decode predictions and labels
    # Make decoding more robust to potentially invalid token IDs from
a poorly trained model

    decoded_preds = []
    for pred_seq in preds:
        if isinstance(pred_seq, (np.ndarray, torch.Tensor)):
            pred_seq_list = pred_seq.tolist()
        else:
            pred_seq_list = list(pred_seq) # Ensure it's a list

        valid_pred_tokens = []
        for token_id in pred_seq_list:
            if isinstance(token_id, int) and 0 <= token_id <
tokenizer.vocab_size:
                valid_pred_tokens.append(token_id)
            else:
                pass

        if not valid_pred_tokens: # If all tokens were invalid, or
sequence became empty
            decoded_preds.append("")
        else:
            try:

decoded_preds.append(tokenizer.decode(valid_pred_tokens,
skip_special_tokens=True))
            except IndexError:
                decoded_preds.append("")
    # Decode labels (usually less problematic)
    decoded_labels = tokenizer.batch_decode(labels,
skip_special_tokens=True)


    decoded_labels = [label if label else "<empty>" for label in
decoded_labels]
    decoded_preds = [pred if pred else "<empty_pred>" for pred in
decoded_preds]


    # Calculate Word Error Rate (WER)
    try:
        wer = wer_metric.compute(predictions=decoded_preds,
references=decoded_labels)
    except Exception as e:
        print(f"Error computing WER: {e}. Decoded preds:
{decoded_preds[:3]}, Decoded labels: {decoded_labels[:3]}")
        wer = 1.0 # Assign a bad score

    # Calculate Character Error Rate (CER)
    try:
        cer = cer_metric.compute(predictions=decoded_preds,
references=decoded_labels)
    except Exception as e:
        print(f"Error computing CER: {e}. Decoded preds:
{decoded_preds[:3]}, Decoded labels: {decoded_labels[:3]}")
        cer = 1.0 # Assign a bad score

    # Calculate exact match accuracy
    exact_matches = sum([1 if pred == label else 0 for pred, label in
zip(decoded_preds, decoded_labels)])
    exact_match_accuracy = exact_matches / len(decoded_preds) if
len(decoded_preds) > 0 else 0.0

    return {
        "wer": wer,
        "cer": cer,
        "exact_match": exact_match_accuracy
    }

# 6. Load and configure the model
print(f"Loading {MODEL_NAME} model...")
model = T5ForConditionalGeneration.from_pretrained(MODEL_NAME,
cache_dir=CACHE_DIR)

model.to(device)  # Move model to GPU if available

# Print model size for reference
```

```
model_size = sum(p.numel() for p in model.parameters()) / 1_000_000
print(f"Model size: {model_size:.2f}M parameters")
```

```
Loading t5-base model...
Model size: 222.90M parameters
```

```
# 7. Setup training arguments
# For extreme memory efficiency on Kaggle T4x2, use these settings:
batch_size = 8  # Reduced from 16
gradient_accumulation_steps = 2  # Increased from 2

# Disable wandb to avoid initialization issues
os.environ["WANDB_DISABLED"] = "true"

training_args = Seq2SeqTrainingArguments(
    output_dir=OUTPUT_DIR,
    eval_strategy="steps",
    eval_steps=50,  # Evaluate more frequently since we have less data
    save_strategy="steps",
    save_steps=50,  # Save more frequently
    save_total_limit=1,  # Keep only the 1 most recent checkpoint to
save space
    learning_rate=5e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    weight_decay=0.01,
    gradient_accumulation_steps=gradient_accumulation_steps,
    num_train_epochs=5,
    predict_with_generate=True,
    generation_max_length=MAX_TARGET_LENGTH,
    fp16=True,  # Use mixed precision for memory efficiency
    logging_dir=os.path.join(OUTPUT_DIR, "logs"),
    logging_steps=20,  # Log more frequently
    load_best_model_at_end=True,
    metric_for_best_model="cer",  # Use CER as the metric to track for
best model
    greater_is_better=False,  # Lower CER is better
    remove_unused_columns=True,
    dataloader_pin_memory=False,  # Disable to save memory
    optim="adamw_torch",  # Use Adafactor optimizer instead of AdamW
(uses less memory)
    report_to=["none"],  # Disable all reporting to save memory and
avoid issues
)

# Data collator
data_collator = DataCollatorForSeq2Seq(
    tokenizer=tokenizer,
    model=model,
    label_pad_token_id=-100,
    pad_to_multiple_of=8 if training_args.fp16 else None
)

# 8. Initialize trainer with early stopping to prevent overfitting
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3)]
)

# Add debugging info before training
print("\nVerifying datasets before training:")
print(f"Training examples: {len(train_dataset)}")
print(f"Validation examples: {len(val_dataset)}")
print(f"Test examples: {len(test_dataset)}")

# Check one example from training data
example = train_dataset[0]
print("\nExample input features:")
print(f"- input_ids shape: {len(example['input_ids'])}")
print(f"- attention_mask shape: {len(example['attention_mask'])}")
print(f"- labels shape: {len(example['labels'])}")

# Free some memory before training
gc.collect()
torch.cuda.empty_cache()
```

```
Verifying datasets before training:
Training examples: 1962
Validation examples: 1970
Test examples: 1876

Example input features:
- input_ids shape: 64
- attention_mask shape: 64
- labels shape: 64

/tmp/ipykernel_1893/1885090836.py:2: FutureWarning: `tokenizer` is
deprecated and will be removed in version 5.0.0 for
`Seq2SeqTrainer.__init__`. Use `processing_class` instead.
  trainer = Seq2SeqTrainer(
```

```
# 9. Train the model
print("\nStarting model training with ultra-light settings...")
trainer.train()
```

```
# Clear GPU memory
gc.collect()
torch.cuda.empty_cache()
```

```
Starting model training with ultra-light settings...

Passing a tuple of `past_key_values` is deprecated and will be removed
in Transformers v4.48.0. You should pass an instance of
`EncoderDecoderCache` instead, e.g.
`past_key_values=EncoderDecoderCache.from_legacy_cache(past_key_values
)`.
/usr/local/lib/python3.11/dist-packages/torch/nn/parallel/_functions.p
y:70: UserWarning: Was asked to gather along dimension 0, but all
input tensors were scalars; will instead unsqueeze and return a
vector.
  warnings.warn(

<IPython.core.display.HTML object>

/usr/local/lib/python3.11/dist-packages/torch/nn/parallel/
_functions.py:70: UserWarning: Was asked to gather along dimension 0,
but all input tensors were scalars; will instead unsqueeze and return
a vector.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torch/nn/parallel/_functions.p
y:70: UserWarning: Was asked to gather along dimension 0, but all
input tensors were scalars; will instead unsqueeze and return a
vector.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torch/nn/parallel/_functions.p
y:70: UserWarning: Was asked to gather along dimension 0, but all
input tensors were scalars; will instead unsqueeze and return a
vector.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torch/nn/parallel/_functions.p
y:70: UserWarning: Was asked to gather along dimension 0, but all
input tensors were scalars; will instead unsqueeze and return a
vector.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torch/nn/parallel/_functions.p
y:70: UserWarning: Was asked to gather along dimension 0, but all
input tensors were scalars; will instead unsqueeze and return a
vector.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torch/nn/parallel/_functions.p
y:70: UserWarning: Was asked to gather along dimension 0, but all
input tensors were scalars; will instead unsqueeze and return a
vector.
```

```
  warnings.warn(
There were missing keys in the checkpoint model loaded:
['encoder.embed_tokens.weight', 'decoder.embed_tokens.weight',
'lm_head.weight'].
```

```
# 10. Evaluate the model on the test set
print("Evaluating model on test set...")
test_results = trainer.evaluate(test_dataset)
print(f"Test results: {test_results}")
```

```
Evaluating model on test set...
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/parallel/
_functions.py:70: UserWarning: Was asked to gather along dimension 0,
but all input tensors were scalars; will instead unsqueeze and return
a vector.
  warnings.warn(
```

```
<IPython.core.display.HTML object>
```

```
Test results: {'eval_loss': 0.1557590812444687, 'eval_wer':
0.08841262157285569, 'eval_cer': 0.04734651833539349,
'eval_exact_match': 0.18550106609808104, 'eval_runtime': 150.032,
'eval_samples_per_second': 12.504, 'eval_steps_per_second': 0.786,
'epoch': 4.926829268292683}
```

```
# 11. Save the model
trainer.save_model(os.path.join(OUTPUT_DIR, "best_model"))
print(f"Best model saved to {os.path.join(OUTPUT_DIR, 'best_model')}")
```

```
Best model saved to /kaggle/working/outputs/best_model
```

```
print("Manually loading the last checkpoint...")
# List contents of output_dir to find the last checkpoint
checkpoint_folders = [f for f in os.listdir(OUTPUT_DIR) if
f.startswith("checkpoint-") and os.path.isdir(os.path.join(OUTPUT_DIR,
f))]
if checkpoint_folders:
    # Sort them to find the one with the highest step number
    checkpoint_folders.sort(key=lambda x: int(x.split('-')[1]))
    latest_checkpoint_dir = os.path.join(OUTPUT_DIR,
checkpoint_folders[-1])
    print(f"Found latest checkpoint: {latest_checkpoint_dir}")

    # Load the model from this checkpoint
    # Make sure 'model' variable is reassigned
    try:
        model =
T5ForConditionalGeneration.from_pretrained(latest_checkpoint_dir)
        model.to(device) # Move to device again
        print("Model successfully loaded manually from the latest
checkpoint.")
        # If you also saved tokenizer and config, you might load them
too, but usually not needed if tokenizer is already loaded.
    except Exception as e:
        print(f"Error loading model manually: {e}")
        # This will show if the checkpoint itself is corrupted or has
missing keys
else:
    print("No checkpoint folders found in output directory.")
```

```
Manually loading the last checkpoint...
Found latest checkpoint: /kaggle/working/outputs/checkpoint-300
Model successfully loaded manually from the latest checkpoint.
```

```
# 12. Function to correct spelling of new sentences
def correct_spelling(sentences):
    inputs = ["correct spelling: " + text for text in sentences]
    inputs = tokenizer(inputs, return_tensors="pt", padding=True,
truncation=True, max_length=MAX_SOURCE_LENGTH)
    inputs = {k: v.to(device) for k, v in inputs.items()}

    outputs = model.generate(
        **inputs,
        max_length=MAX_TARGET_LENGTH,
        num_beams=4,
        early_stopping=True
    )

    corrected = tokenizer.batch_decode(outputs,
skip_special_tokens=True)
    return corrected

def correct_spelling(sentences):

    print(f"Input to correction function: {sentences[:2]}")

    inputs = ["correct spelling: " + text for text in sentences]

    # Tokenize
    input_tokenized = tokenizer(
        inputs,
        return_tensors="pt",
        padding=True,
        truncation=True,
        max_length=MAX_SOURCE_LENGTH
    )

    input_tokenized = {k: v.to(device) for k, v in
input_tokenized.items()}
```

```
    print(f"Input shapes: {input_tokenized['input_ids'].shape}")

    with torch.no_grad():
        outputs = model.generate(
            input_ids=input_tokenized['input_ids'],
            attention_mask=input_tokenized['attention_mask'],
            max_length=MAX_TARGET_LENGTH,
            num_beams=2,
            early_stopping=True,
            do_sample=False,  # Disable sampling for more
deterministic output
            length_penalty=1.0,  # Neutral length penalty
            min_length=1,  # Ensure at least 1 token is generated
            no_repeat_ngram_size=2  # Prevent repetition
        )

    print(f"Output shape: {outputs.shape}")
    print(f"Raw output tokens: {outputs[0][:10].tolist()}")  # Print
first few tokens

    # Decode outputs with more visibility into the process
    corrected = tokenizer.batch_decode(outputs,
skip_special_tokens=True)

    # If still empty, try decoding without skipping special tokens
    if all(not text for text in corrected):
        print("Warning: Empty decoded outputs, trying alternative
decoding...")
        corrected = tokenizer.batch_decode(outputs,
skip_special_tokens=False)
        # Remove only the most common special tokens manually
        corrected = [text.replace("<pad>", "").replace("</s>",
"").replace("<s>", "").strip()
                    for text in corrected]

    print(f"Decoded outputs: {corrected[:2]}")

    return corrected

print("Pad token:", tokenizer.pad_token)
print("EOS token:", tokenizer.eos_token)
print("Decoder start token:", tokenizer.pad_token_id)  # T5 often uses
pad as decoder start
```

```
Pad token: <pad>
EOS token: </s>
Decoder start token: 0
```

```
# 13. Demonstrate the model on a few examples
print("\nTesting the model on some examples:")
```

```python
test_examples = test_data.iloc[6:11].copy()

print("Original vs. Misspelled vs. Corrected:")
for _, row in test_examples.iterrows():
    misspelled = row['misspelled']
    correct = row['correct']

    # Add try-except to catch any errors
    try:
        corrected = correct_spelling([misspelled])[0]
    except Exception as e:
        print(f"Error during correction: {e}")
        corrected = ""  # Default to empty string on error

    print(f"Misspelled: {misspelled}")
    print(f"Correct   : {correct}")
    print(f"Corrected : {corrected}")

    # Calculate and show edit distance between corrected and correct
versions
    if corrected:
        edit_dist = edit_distance(corrected, correct)
        print(f"Edit distance (corrected vs correct): {edit_dist}\n")
    else:
        print("Couldn't calculate edit distance (empty correction)\n")
```

```
Testing the model on some examples:
Original vs. Misspelled vs. Corrected:
Input to correction function: ['Due to my involvement inthe yachting
industry , I am quihe knowledgeable in aisisting in thid industry as
well as most other marine related information .']
Input shapes: torch.Size([1, 41])
Output shape: torch.Size([1, 32])
Raw output tokens: [0, 6984, 12, 82, 9683, 16, 8, 18082, 53, 681]
Decoded outputs: ['Due to my involvement in the yachting industry , I
am very knowledgeable in working in this industry as well as most
other marine related information .']
Misspelled: Due to my involvement inthe yachting industry , I am quihe
knowledgeable in aisisting in thid industry as well as most other
marine related information .
Correct   : Due to my involvement in the yachting industry , I am
quite knowledgeable in assisting in this industry as well as most
other marine related information .
Corrected : Due to my involvement in the yachting industry , I am very
knowledgeable in working in this industry as well as most other marine
related information .
Edit distance (corrected vs correct): 11

Input to correction function: ['Contin uing no rth to northeast LA 27
enters DeiRdder as South Pine Street andx intersects with US 71/US 190
at teh northenr terinus .']
Input shapes: torch.Size([1, 54])
Output shape: torch.Size([1, 37])
Raw output tokens: [0, 3, 16798, 76, 53, 3457, 12, 25806, 5292, 2307]
Decoded outputs: ['Continuing north to northeast LA 27 enters Downtown
as South Pine Street and intersects with US 71/US 190 at the north end
of downtown .']
Misspelled: Contin uing no rth to northeast LA 27 enters DeiRdder as
South Pine Street andx intersects with US 71/US 190 at teh northenr
terinus .
Correct   : Continuing north to northeast LA 27 enters DeRidder as
South Pine Street and intersects with US 171/US 190 at the northern
terminus .
Corrected : Continuing north to northeast LA 27 enters Downtown as
South Pine Street and intersects with US 71/US 190 at the north end of
downtown .
Edit distance (corrected vs correct): 21

Input to correction function: ["Sutati Veerabhadra Rao ,popular as one
of the famous '' Suti Jnta '' , was a popular Radio and TheateArtist
from 1970 - 1980 ."]
Input shapes: torch.Size([1, 53])
Output shape: torch.Size([1, 50])
Raw output tokens: [0, 1923, 7810, 3901, 15, 7093, 1024, 3515, 2922,
32]
Decoded outputs: ["Sutati Veerabhadra Rao ,popular as one of the
famous '' Suti Jnta ’' ; was a popular Radio and Theate Artist from
1970 - 1980 ."]
Misspelled: Sutati Veerabhadra Rao ,popular as one of the famous ''
Suti Jnta '' , was a popular Radio and TheateArtist from 1970 - 1980 .
Correct   : Sutti Veerabhadra Rao , popular as one of the famous ''
Sutti Janta '' , was a popular Radio and Theater Artist from 1970 -
1980 .
Corrected : Sutati Veerabhadra Rao ,popular as one of the famous ''
Suti Jnta ’' ; was a popular Radio and Theate Artist from 1970 -
1980 .
Edit distance (corrected vs correct): 7

Input to correction function: ["She is a popularsinger wh is co
nsidered a idol in Japan during he 1970 's , and is thought to be
representative of that era ."]
Input shapes: torch.Size([1, 44])
Output shape: torch.Size([1, 36])
Raw output tokens: [0, 451, 19, 3, 9, 1012, 7634, 113, 19, 1702]
Decoded outputs: ["She is a popular singer who is considered  an idol
in Japan during the 1970 's , and is thought to be representative of
that era ."]
Misspelled: She is a popularsinger wh is co nsidered a idol in Japan
```

```
during he 1970 's , and is thought to be representative of that era .
Correct   : She is a popular singer who is considered a idol in Japan
during the 1970 's , and is thought to be representative of that era .
Corrected : She is a popular singer who is considered  an idol in
Japan during the 1970 's , and is thought to be representative of that
era .
Edit distance (corrected vs correct): 2

Input to correction function: ['IMDb In the early 1990s , Frederik van
Pa llandt settled in the Philippines where he and his Filipino
girlfriend , Susannah , wezre shot dead in 1994 .']
Input shapes: torch.Size([1, 49])
Output shape: torch.Size([1, 44])
Raw output tokens: [0, 27, 11731, 115, 86, 8, 778, 5541, 7, 3]
Decoded outputs: ['IMDb In the early 1990s , Frederik van Paaland
settled in the Philippines where he and his Filipino girlfriend -
Susannah   -- were shot dead in 1994 .']
Misspelled: IMDb In the early 1990s , Frederik van Pa llandt settled
in the Philippines where he and his Filipino girlfriend , Susannah ,
wezre shot dead in 1994 .
Correct   : IMDb In the early 1990s , Frederik van Pallandt settled in
the Philippines where he and his Filipino girlfriend , Susannah , were
shot dead in 1994 .
Corrected : IMDb In the early 1990s , Frederik van Paaland settled in
the Philippines where he and his Filipino girlfriend - Susannah   --
were shot dead in 1994 .
Edit distance (corrected vs correct): 7
```