```
!pip install -q transformers datasets peft bitsandbytes accelerate
sentencepiece

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.4/59.4 MB 46.4 MB/s eta
0:00:00

import pandas as pd
import torch
from datasets import Dataset
from transformers import (
    AutoTokenizer,
    AutoModelForSeq2SeqLM,
    DataCollatorForSeq2Seq,
    Seq2SeqTrainingArguments,
    Seq2SeqTrainer,
)

MODEL_ID = "t5-small"
OUTPUT_DIR = "./singlish_translator"

FILE_PATH = "singlish_to_english_huggingface.csv"
SOURCE_COL = "singlish"
TARGET_COL = "english"

print("Loading dataset...")
df = pd.read_csv(FILE_PATH)
print(f"Loaded {len(df)} rows")

# Clean data
df[SOURCE_COL] = df[SOURCE_COL].astype(str).str.strip()
df[TARGET_COL] = df[TARGET_COL].astype(str).str.strip()
df = df.dropna().drop_duplicates()

print(f"After cleaning: {len(df)} rows")
print("\nSample:")
print(df.head(3).to_string())

# Shuffle
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

# Split
dataset = Dataset.from_pandas(df)
dataset = dataset.train_test_split(test_size=0.15, seed=42)

print(f"\nTrain: {len(dataset['train'])} | Val:
{len(dataset['test'])}")

print("\nLoading tokenizer...")
tokenizer = AutoTokenizer.from_pretrained(MODEL_ID, legacy=False)

def preprocess_function(examples):
    inputs = ["translate Singlish to English: " + text for text in
examples[SOURCE_COL]]
    targets = examples[TARGET_COL]

    model_inputs = tokenizer(
        inputs,
        max_length=128,
        truncation=True,
        padding=False
    )

    # Tokenize targets
    labels = tokenizer(
        text_target=targets,
        max_length=128,
        truncation=True,
        padding=False
    )

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

print("Tokenizing...")
tokenized_datasets = dataset.map(
    preprocess_function,
    batched=True,
    remove_columns=dataset["train"].column_names,
    desc="Tokenizing"
)

print("\nLoading model...")
model = AutoModelForSeq2SeqLM.from_pretrained(MODEL_ID)


device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Using device: {device}")

training_args = Seq2SeqTrainingArguments(
    output_dir=OUTPUT_DIR,
    eval_strategy="epoch",
    save_strategy="epoch",
    learning_rate=3e-4,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=10,
    weight_decay=0.01,
    save_total_limit=2,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    predict_with_generate=True,
```

```python
    generation_max_length=128,
    generation_num_beams=4,
    fp16=torch.cuda.is_available(),
    logging_steps=50,
    warmup_steps=500,
    report_to="none",
    push_to_hub=False,
)

data_collator = DataCollatorForSeq2Seq(
    tokenizer=tokenizer,
    model=model,
    padding=True
)
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
)

print("\nStarting training...")
print("This will take 10-20 minutes depending on your GPU\n")

trainer.train()

print("\nSaving model...")
trainer.save_model("./singlish_translator_final")
tokenizer.save_pretrained("./singlish_translator_final")
print("Model saved!")

print("\n" + "="*70)
print("TESTING MODEL")
print("="*70)

test_cases = [
    "Wah lau eh, why the queue so long?",
    "I go makan first, you want dapao anything?",
    "Eh, you know how to chop the garlic not?",
    "Later you heat up the oil then you throw in the onions can
already.",
    "Wah, the chicken you marinate until so shiok!",
    "Aiyah, I forget to buy the fish sauce, how?",
]

expected = [
    "Oh my goodness, why is the queue so long?",
    "I'll go eat first, do you want me to get takeaway for anything?",
    "Hey, do you know how to chop the garlic?",
    "Later, you heat up the oil and then you can add the onions.",
    "Wow, the chicken you've marinated tastes amazing!",
    "Oh no, I forgot to buy the fish sauce, what should I do?",
]

model.eval()
model.to(device)

for i, singlish_text in enumerate(test_cases):
    input_text = f"translate Singlish to English: {singlish_text}"

    inputs = tokenizer(
        input_text,
        return_tensors="pt",
        max_length=128,
        truncation=True
    ).to(device)

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=128,
            num_beams=4,
            early_stopping=True,
            no_repeat_ngram_size=3,
            temperature=1.0,
        )

    translation = tokenizer.decode(outputs[0],
skip_special_tokens=True)

    print(f"\n[Test {i+1}]")
    print(f"Singlish:  {singlish_text}")
    print(f"Output:    {translation}")
    print(f"Expected:  {expected[i]}")
    print("-" * 70)

print("\nTesting complete!")
print("\nIf translations are still poor:")
print("   1. Check if your CSV has good quality English translations")
print("   2. Make sure you have 500+ diverse examples")
print("   3. Try training for more epochs (increase
num_train_epochs)")
print("   4. Consider using t5-base instead of t5-small")
```

```
Loading dataset...
 Loaded 1145 rows
After cleaning: 1145 rows
```

```
□ Sample:
   index
singlish                                                               english
0      1                             Eh, you know how to chop the
garlic not?                     Hey, do you know how to chop the
garlic?
1      2  Later you heat up the oil then you throw in the onions can
already.  Later, you heat up the oil and then you can add the onions.
2      3                        Wah, the chicken you marinate until so
shiok!            Wow, the chicken you've marinated tastes amazing!

□ Train: 973 | Val: 172

□ Loading tokenizer...

/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 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":"70afdfa70faf404d8831c617eee6b084","version_major":2,"vers
ion_minor":0}

{"model_id":"36de197a3284475ebfef0b6795d2e59a","version_major":2,"vers
ion_minor":0}

{"model_id":"72d85548b93a41c2ad280cb8d17979b9","version_major":2,"vers
ion_minor":0}

□ Tokenizing...

{"model_id":"c8a92d3310234d0e92a1dfa450db8250","version_major":2,"vers
ion_minor":0}

{"model_id":"2bea60f5ebf3435f847eeb39755fcbb5","version_major":2,"vers
ion_minor":0}


🤖 Loading model...

{"model_id":"ac30c31a737e4a9886b7fd23311d7bfa","version_major":2,"vers
ion_minor":0}

{"model_id":"17e32f357375441997303f2f96b53081","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"410335996713433 3a6eacb34e15bfb73","version_major":2,"version_minor":0}
```

```
Using device: cuda
```

```
/tmp/ipython-input-1456589460.py:131: FutureWarning: `tokenizer` is
deprecated and will be removed in version 5.0.0 for
`Seq2SeqTrainer.__init__`. Use `processing_class` instead.
  trainer = Seq2SeqTrainer(
```

```
 Starting training...
This will take 10-20 minutes depending on your GPU
```

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

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

```
 Saving model...
 Model saved!

======================================================================
 TESTING MODEL
======================================================================

[Test 1]
Singlish:  Wah lau eh, why the queue so long?
Output:    Wow, why is the queue so long?
Expected:  Oh my goodness, why is the queue so long?
----------------------------------------------------------------------

[Test 2]
Singlish:  I go makan first, you want dapao anything?
Output:    I'm going to makan first, do you want dapao anything?
Expected:  I'll go eat first, do you want me to get takeaway for
anything?
----------------------------------------------------------------------

[Test 3]
Singlish:  Eh, you know how to chop the garlic not?
Output:    Hey, do you know how to chop the garlic?
Expected:  Hey, do you know how to chop the garlic?
----------------------------------------------------------------------

[Test 4]
Singlish:  Later you heat up the oil then you throw in the onions can
already.
```

```
Output:    Later, you heat up the oil and then you can add the onions.
Expected:  Later, you heat up the oil and then you can add the onions.
----------------------------------------------------------------------

[Test 5]
Singlish:  Wah, the chicken you marinate until so shiok!
Output:    Wow, the chicken you've marinated is so delicious!
Expected:  Wow, the chicken you've marinated tastes amazing!
----------------------------------------------------------------------

[Test 6]
Singlish:  Aiyah, I forget to buy the fish sauce, how?
Output:    Oh no, I forgot to buy the fish sauce, what should I do?
Expected:  Oh no, I forgot to buy the fish sauce, what should I do?
----------------------------------------------------------------------

□ Testing complete!

□ If translations are still poor:
   1. Check if your CSV has good quality English translations
   2. Make sure you have 500+ diverse examples
   3. Try training for more epochs (increase num_train_epochs)
   4. Consider using t5-base instead of t5-small
```

```python
import pandas as pd
import torch
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
from tqdm import tqdm

print("□ Loading trained model...")
MODEL_PATH = "./singlish_translator_final"

try:
    tokenizer = AutoTokenizer.from_pretrained(MODEL_PATH)
    model = AutoModelForSeq2SeqLM.from_pretrained(MODEL_PATH)
    print("Model loaded successfully!")
except Exception as e:
    print(f"Error loading model: {e}")
    print("Make sure you've trained the model first!")
    raise

device = "cuda" if torch.cuda.is_available() else "cpu"
model.to(device)
model.eval()
print(f"Using device: {device}")

print("\n□ Loading test dataset...")
TEST_FILE = "singlish_hetavi_refined.csv"

try:
    df = pd.read_csv(TEST_FILE)
    # Strip whitespace from column names
    df.columns = df.columns.str.strip()
    print(f"Loaded {len(df)} rows from {TEST_FILE}")
    print(f"Columns: {df.columns.tolist()}")

    # Check for required columns
    if 'singlish' not in df.columns or 'english' not in df.columns:
        raise ValueError("CSV must have 'singlish' and 'english'
columns!")

    # Clean data
    df['singlish'] = df['singlish'].astype(str).str.strip()
    df['english'] = df['english'].astype(str).str.strip()

    print("\nSample data:")
    print(df.head(3))

except FileNotFoundError:
    print(f"File '{TEST_FILE}' not found!")
    print("Make sure the file is in the same directory as this
notebook.")
    raise
except Exception as e:
    print(f"Error loading file: {e}")
    raise

print(f"\n□ Translating {len(df)} Singlish sentences...")
print("This may take a few minutes...\n")

translations = []

for idx, row in tqdm(df.iterrows(), total=len(df),
desc="Translating"):
    singlish_text = row['singlish']

    input_text = f"translate Singlish to English: {singlish_text}"

    # Tokenize
    inputs = tokenizer(
        input_text,
        return_tensors="pt",
        max_length=128,
        truncation=True
    ).to(device)

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=128,
            num_beams=4,
            early_stopping=True,
            no_repeat_ngram_size=3,
            temperature=1.0,
        )

    # Decode
    translation = tokenizer.decode(outputs[0],
skip_special_tokens=True)
    translations.append(translation)

print("\nCreating output file...")

output_df = pd.DataFrame({
    'singlish': df['singlish'],
    'translated_output': translations,
    'target_output': df['english']
})

OUTPUT_FILE = "singlish_translation_results.csv"
output_df.to_csv(OUTPUT_FILE, index=False)

print(f"Results saved to '{OUTPUT_FILE}'")

print("\n" + "="*80)
print("SAMPLE RESULTS (First 10 rows)")
print("="*80)

for idx in range(min(10, len(output_df))):
    row = output_df.iloc[idx]
    print(f"\n[Row {idx + 1}]")
    print(f"Singlish:    {row['singlish']}")
    print(f"Translated:  {row['translated_output']}")
    print(f"Target:      {row['target_output']}")
    print("-" * 80)

print("\n" + "="*80)
print("QUALITY ANALYSIS")
print("="*80)

exact_matches = sum(
    output_df['translated_output'].str.lower() ==
output_df['target_output'].str.lower()
)

print(f"\nExact matches: {exact_matches}/{len(output_df)}
({exact_matches/len(output_df)*100:.1f} %)")

avg_singlish_len = output_df['singlish'].str.len().mean()
avg_translated_len = output_df['translated_output'].str.len().mean()
```

```
avg_target_len = output_df['target_output'].str.len().mean()

print(f"\nAverage character lengths:")
print(f"   Singlish:    {avg_singlish_len:.1f} chars")
print(f"   Translated:  {avg_translated_len:.1f} chars")
print(f"   Target:      {avg_target_len:.1f} chars")

empty_translations = sum(output_df['translated_output'].str.strip() ==
'')
print(f"\nEmpty translations: {empty_translations}")

copies = sum(
    output_df['singlish'].str.lower() ==
output_df['translated_output'].str.lower()
)
print(f"Direct copies (no translation): {copies}")

print("\nAnalysis complete!")
print(f"\nFull results saved to: {OUTPUT_FILE}")
print("You can open this CSV file to review all translations")

□ Loading trained model...
□ Model loaded successfully!
  Using device: cuda

□ Loading test dataset...
□ Loaded 300 rows from singlish_hetavi_refined.csv
□ Columns: ['singlish', 'english']

□ Sample data:
                                            singlish  \
0          Wah, the kopi today taste damn solid lah!
1  Walao… she tell me she don't love me anymore… ...
2    Alamak, the dog run off from backyard just now!

                                             english
0          Wow, the coffee today tastes really good!
1  She told me she doesn't love me anymore… my he...
2  Oh no, the dog ran off from the backyard just ...

□ Translating 300 Singlish sentences...
This may take a few minutes...


Translating: 100%|██████████| 300/300 [01:14<00:00,  4.02it/s]


□ Creating output file...
□ Results saved to 'singlish_translation_results.csv'

======================================================================
==========
□ SAMPLE RESULTS (First 10 rows)
======================================================================
==========

[Row 1]
Singlish:    Wah, the kopi today taste damn solid lah!
Translated:  Wow, the coffee is really good today!
Target:      Wow, the coffee today tastes really good!
----------------------------------------------------------------------
----------

[Row 2]
Singlish:    Walao… she tell me she don't love me anymore… my heart
really pain sia.
Translated:  Walao... she told me she doesn't love me anymore... my
heart is really painless.
Target:      She told me she doesn't love me anymore… my heart really
hurts.
----------------------------------------------------------------------
----------

[Row 3]
Singlish:    Alamak, the dog run off from backyard just now!
Translated:  Alamak, the dog was running off from the backyard just
now!
Target:      Oh no, the dog ran off from the backyard just now!
----------------------------------------------------------------------
----------

[Row 4]
Singlish:    I takut I lose my job if this target cannot hit one.
Translated:  I'm afraid I'll lose my job if this target isn't hit.
Target:      I'm afraid I'll lose my job if I can't hit this target.
----------------------------------------------------------------------
----------

[Row 5]
Singlish:    He keep talking cock during meeting lor, cannot tahan.
Translated:  He keeps talking cockily during the meeting, I can't
stand it.
Target:      He keeps talking nonsense during the meeting, can't stand
it.
----------------------------------------------------------------------
----------

[Row 6]
Singlish:    Eh, you coming to makan later or not?
Translated:  Hey, are you coming to makan later?
Target:      Hey, are you coming to eat later or not?
```

```
--------------------------------------------------------------------------------

[Row 7]
Singlish:    Yesterday we lepak by the beach until sunset, damn
relaxing lah.
Translated:  Yesterday we walked by the beach until the sunset; it was
really relaxing.
Target:      Yesterday we hung out by the beach until sunset, very
relaxing.
--------------------------------------------------------------------------------

[Row 8]
Singlish:    The laptop suddenly freeze again, confirm kena virus lor.
Translated:  The laptop suddenly freezes again, it's definitely
causing a virus.
Target:      The laptop suddenly froze again, definitely caught a
virus.
--------------------------------------------------------------------------------

[Row 9]
Singlish:    She received her results, stunned like cannot believe
leh.
Translated:  She received her results – stunned like I can't believe
it.
Target:      She received her results, and was stunned like she
couldn't believe it.
--------------------------------------------------------------------------------

[Row 10]
Singlish:    I still scared to try that roller-coaster ride, no guts
one.
Translated:  I'm still scared to try that roller-coaster ride, there's
no guts.
Target:      I'm still afraid to try that roller-coaster ride, don't
have the guts.
--------------------------------------------------------------------------------

================================================================================
 QUALITY ANALYSIS
================================================================================

✓ Exact matches: 5/300 (1.7 %)
```

```
□ Average character lengths:
   Singlish:    62.6 chars
   Translated:  66.9 chars
   Target:      66.9 chars

⚠  Empty translations: 0
⚠  Direct copies (no translation): 0

□ Analysis complete!

□ Full results saved to: singlish_translation_results.csv
□ You can open this CSV file to review all translations
```