```
#installing dependencies
!pip install --upgrade transformers datasets accelerate sentencepiece
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.53.3)
Collecting transformers
  Downloading transformers-4.54.0-py3-none-any.whl.metadata (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.7/41.7 kB 3.3 MB/s eta 0:00:00
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-packages (4.0.0)
Requirement already satisfied: accelerate in /usr/local/lib/python3.11/dist-packages (1.9.0)
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.11/dist-packages (0.2.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.18.0)
Collecting huggingface-hub<1.0,>=0.34.0 (from transformers)
  Downloading huggingface_hub-0.34.1-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.21.2)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages (from datasets) (3.5.0)
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.11/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2025.3.0,>=2023.1.0 in /usr/local/lib/python3.11/dist-packages (from fsspec[http]<=2025.3.0
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: torch>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from accelerate) (2.6.0+cu124)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.11/dist-packages (from fsspec[http]<=2025.3.0,
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.34.0-
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2025.7
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->accelerate) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->accelerate) (3.1.6)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=2.0.0->accelerate)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=2.0.0->accelerate)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=2.0.0->accelerate)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=2.0.0->accelerate)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=2.0.0->accelerate)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=2.0.0->accelerate)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=2.0.0->accelerate)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=2.0.0->accelerate)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=2.0.0->accelerate)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->accele
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->accelerate)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->accelerat
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=2.0.0->accelerate)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
```

```
import pandas as pd

# Load manually labeled CSV
df = pd.read_csv("https://raw.githubusercontent.com/psabhay2003/NLP-driven-Invoice-Digitalization/refs/heads/main/labeled%20data%20samp

# Create prompt-response style training format
def make_prompt(row):
    return f"Extract invoice fields: {row['extracted_text']}"

def make_output(row):
    return f"Invoice No: {row['Invoice number']}, Date: {row['Date']}, Total Amount: {row['Total Amount']}, Vendor: {row['Vendor']}"

df['input_text'] = df.apply(make_prompt, axis=1)
df['target_text'] = df.apply(make_output, axis=1)
```

```
from datasets import Dataset
from transformers import T5Tokenizer

#tokenization
tokenizer = T5Tokenizer.from_pretrained("t5-small")

train_dataset = Dataset.from_pandas(df[['input_text', 'target_text']])
```

```
def preprocess(example):
    inputs = tokenizer(example['input_text'], max_length=512, truncation=True, padding="max_length")
    targets = tokenizer(example['target_text'], max_length=128, truncation=True, padding="max_length")
    inputs['labels'] = targets['input_ids']
    return inputs

tokenized_dataset = train_dataset.map(preprocess, batched=False)
```

```
/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 :
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(
tokenizer_config.json:  100%                                   2.32k/2.32k  [00:00<00:00,  99.0kB/s]
spiece.model:  100%                                            792k/792k  [00:00<00:00,  1.18MB/s]
tokenizer.json:  100%                                          1.39M/1.39M  [00:00<00:00,  6.04MB/s]
You are using the default legacy behaviour of the <class 'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is expected, ar
Map:  100%                                                     200/200  [00:00<00:00,  769.35  examples/s]
```

```
from transformers import T5ForConditionalGeneration, Seq2SeqTrainingArguments, Seq2SeqTrainer, DataCollatorForSeq2Seq

#fine-tuning the T5 model
model = T5ForConditionalGeneration.from_pretrained("t5-small")

training_args = Seq2SeqTrainingArguments(
    output_dir="./t5_invoice_model",
    per_device_train_batch_size=4,
    gradient_accumulation_steps=2,
    learning_rate=1e-4,
    num_train_epochs=8,
    eval_strategy="steps",
    eval_steps=50,
    load_best_model_at_end=True,
    metric_for_best_model="loss",
    save_total_limit=2,
    fp16=True,
    logging_steps=10,
    report_to="none"
)

data_collator = DataCollatorForSeq2Seq(tokenizer, model=model, label_pad_token_id=-100)

trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    eval_dataset=tokenized_dataset.train_test_split(test_size=0.1)['test'],
    data_collator=data_collator
)

trainer.train()
```

```
config.json:  100%                                   1.21k/1.21k  [00:00<00:00,  120kB/s]
model.safetensors:  100%                             242M/242M  [00:03<00:00,  106MB/s]
generation_config.json:  100%                        147/147  [00:00<00:00,  14.0kB/s]
[200/200 00:54, Epoch 8/8]
```

Step	Training Loss	Validation Loss
50	0.474200	0.289435
100	0.127000	0.057332
150	0.064100	0.040344
200	0.055400	0.037051

```
There were missing keys in the checkpoint model loaded: ['encoder.embed_tokens.weight', 'decoder.embed_tokens.weight', 'lm_head.weig
TrainOutput(global_step=200, training_loss=0.9064843153953552, metrics={'train_runtime': 56.7608, 'train_samples_per_second':
28.188, 'train_steps_per_second': 3.524, 'total_flos': 216546882355200.0, 'train_loss': 0.9064843153953552, 'epoch': 8.0})
```

```
model.save_pretrained("t5-small-model")
tokenizer.save_pretrained("t5-small-model")
```

```
('t5-small-model/tokenizer_config.json',
 't5-small-model/special_tokens_map.json',
        't5-small-model/spiece.model',
        't5-small-model/added_tokens.json')


# Load OCR CSV
ocr_df = pd.read_csv("https://raw.githubusercontent.com/psabhay2003/NLP-driven-Invoice-Management-System/refs/heads/main/invoice_texts.c

# Prepare input prompts
ocr_df['prompt'] = ocr_df['extracted_text'].apply(lambda x: f"Extract invoice fields: {x}")

# Generate predictions for entire range of data
def generate_prediction(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, max_length=512).to(model.device)
    output = model.generate(**inputs, max_length=128)
    return tokenizer.decode(output[0], skip_special_tokens=True)

ocr_df['extracted_fields'] = ocr_df['prompt'].apply(generate_prediction)


ocr_df[['filename', 'extracted_fields']].to_csv("T5_output.csv", index=False)
from google.colab import files
files.download("T5_output.csv")
```

```
# Push this T5 output to github and then convert into the final csv which will be used in SQL database
t5_df = pd.read_csv("https://raw.githubusercontent.com/psabhay2003/NLP-driven-Invoice-Digitalization/refs/heads/main/T5_output.csv")
import re
def parse_fields(txt):
    # Use regex patterns to cover different label styles
    patterns = [
        # Standard “Invoice No: …, Date: …, Total Amount: …, Vendor: …”
        r'Invoice\s*No[:\-]?\s*(?P<inv>[^,;\n]+)[,;\n]\s*Date[:\-]?\s*(?P<date>[^,;\n]+)[,;\n]\s*Total\s*Amount[:\-]?\s*(?P<amt>[^,;\n]+)
        # Using “#” instead of “No”
        r'Invoice\s*#[:\-]?\s*(?P<inv>[^,;\n]+)[,;\n]\s*Date[:\-]?\s*(?P<date>[^,;\n]+)[,;\n]\s*Total[:\-]?\s*(?P<amt>[^,;\n]+)[,;\n]\s*V
        # All four as key:value pairs separated by semicolons
        r'Invoice\s*No[:\-]?\s*(?P<inv>[^;]+);\s*Date[:\-]?\s*(?P<date>[^;]+);\s*Total\s*Amount[:\-]?\s*(?P<amt>[^;]+);\s*Vendor[:\-]?\s*
        # CSV-style “1234,2023-01-01,1500,Acme Corp”
        r'^(?P<inv>[A-Z0-9\-]+)\s*,\s*(?P<date>\d{1,2}[\/\-\.][A-Za-z0-9\/\-\.]+)\s*,\s*(?P<amt>[₹\$]?\s*\d+[.,]?\d*)\s*,\s*(?P<vend>.+)$
    ]
    for pat in patterns:
        m = re.search(pat, txt.strip(), flags=re.IGNORECASE)
        if m:
            return m.group('inv').strip(), m.group('date').strip(), m.group('amt').strip(), m.group('vend').strip()
    # Fallback: split on commas/newlines, then pick by prefix
    parts = re.split(r',|\n|;', txt)
    inv = date = amt = vend = None
    for p in parts:
        p = p.strip()
        low = p.lower()
        if inv is None and 'invoice' in low:
            inv = re.sub(r'[^A-Z0-9\-]', '', p)
        elif date is None and re.search(r'\d{1,2}[\/\-\.\s][A-Za-z]{3,}\s*\d{2,4}', p):
            date = p
        elif amt is None and re.search(r'\d+[.,]?\d*', p):
            amt = p
        elif vend is None and len(p) > 3:
            vend = p
    return inv, date, amt, vend

# Apply parsing
parsed = t5_df['extracted_fields'].apply(lambda x: pd.Series(parse_fields(str(x)),
                                                            index=['Invoice No','Date','Total Amount','Vendor']))
final = pd.concat([t5_df['filename'], parsed], axis=1)

# Save and download
final.to_csv("final_structured_output.csv", index=False)
from google.colab import files
files.download("final_structured_output.csv")
```