```
!pip install datasets
```

Show hidden output

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer, TrainingA
from datasets import Dataset, load_dataset
import pandas as pd
import numpy as np
import torch
import os
import random
import gc
import json
```

```
os.environ["WANDB_DISABLED"] = "true"
os.environ["WANDB_MODE"] = "offline"
```

```
os.environ['CUDA_LAUNCH_BLOCKING'] = "1"
torch.backends.cudnn.benchmark = True
# Check GPU availability and memory
if torch.cuda.is_available():
    device = torch.device("cuda")
    print(f"Using GPU: {torch.cuda.get_device_name(0)}")
    print(f"GPU Memory: {torch.cuda.get_device_properties(0).total_memory / 1e9:.2f}
    gc.collect()
else:
    device = torch.device("cpu")
    print("No GPU available, using CPU instead")
```

```
Using GPU: Tesla T4
GPU Memory: 15.83 GB
```

```
ds = load_dataset("BitAgent/tool_calling_shuffle")
df = pd.DataFrame(ds)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarnin
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
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 mo
  warnings.warn(
README.md: 100%                                                    323/323 [00:00<00:00, 5.90kB/s]
train-00000-                                                       79.9M/79.9M [00:01<00:00, 50.3MB/
of-00001.parquet: 100%                                             s]
Generating train split: 100%                                       551285/551285 [00:04<00:00, 128803.95 examples/
```

```
is_testing=False

if is_testing:
    df = df.head(100)
df
```

| | train |
|---|---|
| **0** | {'conversation': '[{"role": "user", "content":... |
| **1** | {'conversation': '[{"role": "user", "content":... |
| **2** | {'conversation': '[{"role": "user", "content":... |
| **3** | {'conversation': '[{"role": "user", "content":... |
| **4** | {'conversation': '[{"role": "user", "content":... |
| **...** | ... |
| **551280** | {'conversation': '[{"role": "user", "content":... |
| **551281** | {'conversation': '[{"role": "user", "content":... |
| **551282** | {'conversation': '[{"role": "user", "content":... |
| **551283** | {'conversation': '[{"role": "user", "content":... |
| **551284** | {'conversation': '[{"role": "user", "content":... |

551285 rows × 1 columns

```
def extract_content_and_description(conversation, tools):
    """
    Extract user content and tool description from conversation and tools.
    """
    try:
        # Parse JSON strings
        conv_data = json.loads(conversation)
        tools_data = json.loads(tools)

        # Get first user message
        user_content = next(
            turn['content'] for turn in conv_data
            if turn['role'] == 'user'
        )

        # Get first tool's description
        tool_description = tools_data[0]['description'] if tools_data else ''

        return user_content, tool_description
    except (json.JSONDecodeError, KeyError, StopIteration):
        return '', ''
```

```
# Process the data
processed_data = []
for _, row in df.iterrows():
    try:
        data = row['train']

        content, description = extract_content_and_description(
            data['conversation'],
            data['tools']
        )

        if content and description:
            processed_data.append({
                'content': content,
                'description': description
            })
    except (json.JSONDecodeError, KeyError):
        print(f"Error processing row: {row}")
        continue

result_df = pd.DataFrame(processed_data)
result_df
```

| | content | description |
|---|---|---|
| **0** | What was the first named storm of the 2022 Atl... | Returns the name of the first named storm of t... |
| **1** | Delete a service called 'old-service' in the '... | Deletes a service in a given Kubernetes namesp... |
| **2** | Do we have any backorders pending for 'Super D... | Check if there are any backorders for the spec... |
| **3** | What's the 52-week high for Amazon's stock? | Returns the 52-week high for a stock given its... |
| **4** | Are there any impending failures predicted for... | Predicts any impending failures for the specif... |
| **...** | ... | ... |
| **551280** | Please convert this image to grayscale. | Converts the input image to grayscale. |
| **551281** | Execute a command to restart the pod "back-end... | A function to restart a given pod, useful for ... |
| **551282** | When was the last time 'Olivia Thompson' | Returns the date of the last visit for the |

```
label_to_int = {desc: idx for idx, desc in enumerate(result_df['description'].unique(
int_to_label = {idx: desc for desc, idx in label_to_int.items()}
```

```
result_df['label'] = result_df['description'].map(label_to_int)

result_df = result_df.sample(frac=1.0, random_state=42).reset_index(drop=True)
train_size = int(0.8 * len(result_df))
train_df = result_df.iloc[:train_size].reset_index(drop=True)
test_df = result_df.iloc[train_size:].reset_index(drop=True)

train_ds = Dataset.from_pandas(train_df)
test_ds = Dataset.from_pandas(test_df)

num_labels = len(label_to_int)
```

```
print(f"Processed {len(result_df)} examples")
print(f"Number of unique categories: {num_labels}")
print(f"Training set size: {len(train_df)}")
print(f"Test set size: {len(test_df)}")
```

```
Processed 551285 examples
Number of unique categories: 73
Training set size: 441028
Test set size: 110257
```

```
model_name = "facebook/bart-large-mnli"
config = AutoConfig.from_pretrained(model_name)
config.num_labels = num_labels
config.id2label = {i: label for label, i in label_to_int.items()}
config.label2id = label_to_int

# Initialize the model with new config but don't load the classification head
nli_model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    config=config,
    ignore_mismatched_sizes=True
)
```

```
config.json: 100%                                    1.15k/1.15k [00:00<00:00, 83.4kB/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Fall
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the '
model.safetensors: 100%                              1.63G/1.63G [00:08<00:00, 90.7MB/
                                                                                 s]
Some weights of BartForSequenceClassification were not initialized from the model che
- classification_head.out_proj.bias: found shape torch.Size([3]) in the checkpoint an
- classification_head.out_proj.weight: found shape torch.Size([3, 1024]) in the check
You should probably TRAIN this model on a down-stream task to be able to use it for p
```

```
# Initialize tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
tokenizer_config.json: 100%                                    26.0/26.0 [00:00<00:00, 2.40kB/
                                                                                          s]
vocab.json: 100%                                               899k/899k [00:00<00:00, 3.68MB/s]
merges.txt: 100%                                               456k/456k [00:00<00:00, 2.81MB/s]
tokenizer.json: 100%                                           1.36M/1.36M [00:00<00:00, 7.88MB/
```

```
# Split data into train and test sets
# ----------------------

def prepare_dataset_row(row):
    """Prepare a single row of the dataset."""
    try:
        # Parse conversation to get user content
        conversation = json.loads(row['conversation'])
        user_content = next(
            turn['content'] for turn in conversation
            if turn['role'] == 'user'
        )

        # Parse tools to get description
        tools = json.loads(row['tools'])
        description = tools[0]['description'] if tools else ''

        # Only return if both content and description are valid
        if user_content and description:
            return {
                'content': user_content,
                'description': description,
                'label': label_to_int[description]
            }
    except (json.JSONDecodeError, StopIteration, KeyError):
        pass
    return None

# Prepare clean dataset
clean_data = []
for _, row in df.iterrows():
    prepared_row = prepare_dataset_row(row)
    if prepared_row:
        clean_data.append(prepared_row)

# Convert to DataFrame
clean_df = pd.DataFrame(clean_data)

# Shuffle and split the data
train_size = int(0.8 * len(clean_df))
clean_df = clean_df.sample(frac=1, random_state=42).reset_index(drop=True)
df_train = clean_df.iloc[:train_size]
```

```
df_train = clean_df.iloc[:train_size]
df_test = clean_df.iloc[train_size:]

print(f"Original dataset size: {len(df)}")
print(f"Clean dataset size: {len(clean_df)}")
print(f"Data split: {len(df_train)} training samples, {len(df_test)} test samples")
print("\nSample training data:")
df_train.head()
```

```
Original dataset size: 551285
Clean dataset size: 0
Data split: 0 training samples, 0 test samples

Sample training data:
```

```
# Configure training parameters
# ----------------------
print("Configuring training parameters...")
training_args = TrainingArguments(
    output_dir="./results",
    learning_rate=2e-5,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=16,
    num_train_epochs=1,
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=10,
    report_to="none",  # Explicitly disable all external reporting

    # Basic evaluation and saving parameters
    save_steps=500,
    eval_steps=500,

    # Performance optimization
    fp16=True,
    gradient_accumulation_steps=1,
    gradient_checkpointing=True,
    max_grad_norm=1.0,
    dataloader_num_workers=4,
)

```

```
Configuring training parameters...
```

```
def preprocess_function(examples):
    # BART requires proper padding and formatting
    inputs = tokenizer(
        examples["content"],
        padding="max_length"
```

```
        padding="max_length",
        truncation=True,
        max_length=128,
        return_tensors=None,
        return_attention_mask=True,  # Make sure to include attention mask
    )

    # Add labels
    inputs['labels'] = examples['label']

    return inputs

```

```
print("Processing training dataset...")
train_ds = train_ds.map(
    preprocess_function,
    batched=True,
    num_proc=4,
    remove_columns=['content', 'description', 'label'],
    load_from_cache_file=True,
    desc="Tokenizing training data"
)
```

```
Processing training dataset...
Tokenizing training data (num_proc=4): 100%  441028/441028 [01:00<00:00, 1709.47 examples/s]
```

```
print("Processing test dataset...")
test_ds = test_ds.map(
    preprocess_function,
    batched=True,
    num_proc=4,
    remove_columns=['content', 'description', 'label'],
    load_from_cache_file=True,
    desc="Tokenizing test data"
)
```

```
Processing test dataset...
Tokenizing test data (num_proc=4): 100%  110257/110257 [00:14<00:00, 2801.19 examples/s]
```

```

print("Initializing Trainer...")

# Update the trainer with better defaults
trainer = Trainer(
    model=nli_model,
    args=training_args,
    train_dataset=train_ds,
    eval_dataset=test_ds
```

```
)
```

```
Initializing Trainer...
```

```
# Move model to GPU
if torch.cuda.is_available():
    nli_model.to(device)

print("Starting training...")
trainer.train()
```

```
Starting training...
/usr/local/lib/python3.11/dist-packages/torch/utils/data/dataloader.py:624: UserWarni
  warnings.warn(
```

[ 1678/13783 37:09 < 4:28:22, 0.75 it/s, Epoch 0.12/1]

| Step | Training Loss |
|---|---|
| 10 | 2.822100 |
| 20 | 0.801700 |
| 30 | 0.337800 |
| 40 | 0.196400 |
| 50 | 0.139700 |
| 60 | 0.062100 |
| 70 | 0.075500 |
| 80 | 0.066700 |
| 90 | 0.028200 |
| 100 | 0.030000 |
| 110 | 0.040700 |
| 120 | 0.027000 |
| 130 | 0.025900 |
| 140 | 0.025100 |
| 150 | 0.033400 |
| 160 | 0.009800 |
| 170 | 0.017700 |
| 180 | 0.030300 |
| 190 | 0.009900 |
| 200 | 0.019100 |

| | |
|---|---|
| 210 | 0.014600 |
| 220 | 0.022600 |
| 230 | 0.022800 |
| 240 | 0.006400 |
| 250 | 0.009600 |
| 260 | 0.009400 |
| 270 | 0.003800 |
| 280 | 0.003800 |
| 290 | 0.002900 |
| 300 | 0.002500 |
| 310 | 0.002700 |
| 320 | 0.008400 |
| 330 | 0.002300 |
| 340 | 0.002600 |
| 350 | 0.002400 |
| 360 | 0.004600 |
| 370 | 0.002300 |
| 380 | 0.002700 |
| 390 | 0.002200 |
| 400 | 0.001700 |
| 410 | 0.001500 |
| 420 | 0.001700 |
| 430 | 0.018100 |
| 440 | 0.003100 |
| 450 | 0.002000 |
| 460 | 0.001900 |
| 470 | 0.001900 |
| 480 | 0.001900 |
| 490 | 0.001500 |
| 500 | 0.001500 |
| 510 | 0.001900 |

| | |
|---|---|
| 520 | 0.001500 |
| 530 | 0.001100 |
| 540 | 0.001200 |
| 550 | 0.001000 |
| 560 | 0.018000 |
| 570 | 0.002300 |
| 580 | 0.001600 |
| 590 | 0.001200 |
| 600 | 0.001000 |
| 610 | 0.001000 |
| 620 | 0.001000 |
| 630 | 0.001000 |
| 640 | 0.001300 |
| 650 | 0.001100 |
| 660 | 0.000800 |
| 670 | 0.000900 |
| 680 | 0.001100 |
| 690 | 0.001100 |
| 700 | 0.001100 |
| 710 | 0.001000 |
| 720 | 0.000700 |
| 730 | 0.000900 |
| 740 | 0.017400 |
| 750 | 0.001600 |
| 760 | 0.001100 |
| 770 | 0.001000 |
| 780 | 0.001000 |
| 790 | 0.000800 |
| 800 | 0.000800 |
| 810 | 0.000900 |

| | |
|---|---|
| 820 | 0.001000 |
| 830 | 0.000700 |
| 840 | 0.000800 |
| 850 | 0.000600 |
| 860 | 0.000700 |
| 870 | 0.000500 |
| 880 | 0.000800 |
| 890 | 0.000700 |
| 900 | 0.000700 |
| 910 | 0.000800 |
| 920 | 0.000600 |
| 930 | 0.000800 |
| 940 | 0.000600 |
| 950 | 0.000800 |
| 960 | 0.000400 |
| 970 | 0.000500 |
| 980 | 0.016400 |
| 990 | 0.001300 |
| 1000 | 0.001000 |
| 1010 | 0.001100 |
| 1020 | 0.000600 |
| 1030 | 0.000700 |
| 1040 | 0.000700 |
| 1050 | 0.000700 |
| 1060 | 0.000500 |
| 1070 | 0.000600 |
| 1080 | 0.000600 |
| 1090 | 0.000600 |
| 1100 | 0.000500 |
| 1110 | 0.000600 |
| 1120 | 0.000400 |

| | |
|---|---|
| 1130 | 0.000700 |
| 1140 | 0.000500 |
| 1150 | 0.000400 |
| 1160 | 0.000500 |
| 1170 | 0.000600 |
| 1180 | 0.000500 |
| 1190 | 0.000600 |
| 1200 | 0.000500 |
| 1210 | 0.000400 |
| 1220 | 0.000400 |
| 1230 | 0.003600 |
| 1240 | 0.001100 |
| 1250 | 0.001000 |
| 1260 | 0.000700 |
| 1270 | 0.000600 |
| 1280 | 0.000500 |
| 1290 | 0.000500 |
| 1300 | 0.000400 |
| 1310 | 0.000400 |
| 1320 | 0.000400 |
| 1330 | 0.000400 |
| 1340 | 0.007900 |
| 1350 | 0.000900 |
| 1360 | 0.000900 |
| 1370 | 0.000400 |
| 1380 | 0.000500 |
| 1390 | 0.000400 |
| 1400 | 0.000400 |
| 1410 | 0.000400 |
| 1420 | 0.000400 |
| 1430 | 0.000500 |

| | |
|---|---|
| 1440 | 0.000400 |
| 1450 | 0.000400 |
| 1460 | 0.000400 |
| 1470 | 0.000400 |
| 1480 | 0.000300 |
| 1490 | 0.000300 |
| 1500 | 0.000300 |
| 1510 | 0.000300 |
| 1520 | 0.000300 |
| 1530 | 0.000300 |
| 1540 | 0.000400 |
| 1550 | 0.000300 |
| 1560 | 0.000300 |
| 1570 | 0.000300 |
| 1580 | 0.000300 |
| 1590 | 0.000300 |
| 1600 | 0.000400 |
| 1610 | 0.000300 |
| 1620 | 0.000300 |
| 1630 | 0.000400 |
| 1640 | 0.000300 |
| 1650 | 0.000300 |
| 1660 | 0.000300 |
| 1670 | 0.000300 |

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-21-923523ea7b81> in <cell line: 0>()
      4
      5 print("Starting training...")
----> 6 trainer.train()
```

29 frames

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/linear.py in forward(self, input)
```

```
    123
    124     def forward(self, input: Tensor) -> Tensor:
--> 125         return F.linear(input, self.weight, self.bias)
    126
    127     def extra_repr(self) -> str:

KeyboardInterrupt:
```

```
print("Saving model...")
trainer.save_model("./final_model")
print("Training complete!")
```

```
Saving model...
Training complete!
```

```
with open("./label_mapping.json", "w") as f:
    json.dump({"label_to_int": label_to_int, "int_to_label": int_to_label}, f)
print("Label mapping saved to label_mapping.json")
```

```
Label mapping saved to label_mapping.json
```

```
REDACTED
```

```
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.1
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/di
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
Repository ru4en/bart-large-mnli-tool-router-new is ready
Uploading training_args.bin...
training_args.bin: 100%                                   5.24k/5.24k [00:00<00:00, 12.9kB/
                                                                      s]
Uploading config.json...
Uploading model.safetensors...
model.safetensors: 100%                                   1.63G/1.63G [00:56<00:00, 24.6MB/
                                                                      s]
```