

```

# =====
# Project: IntelliAssist - A Generative AI-based IT Service Desk Assistant
# Version 2.0: Debugged and robust fine-tuning script.
# This script addresses the issues from the previous version, ensuring the
# model correctly learns and generates domain-specific responses.
# =====

# -----
# 1. Setup: Install and import necessary libraries
# -----
# This cell needs to be run in Google Colab to install the Hugging Face
# `transformers`, `datasets`, and `accelerate` libraries.
!pip install transformers datasets accelerate --quiet

import torch
from transformers import T5ForConditionalGeneration, T5Tokenizer, Trainer, TrainingArguments
from datasets import Dataset
import pandas as pd
import re # We will use regex for more robust parsing
from IPython.display import display, Markdown

print("Libraries installed and imported successfully.")

# Check for GPU availability
device = 'cuda' if torch.cuda.is_available() else 'cpu'
print(f"Using device: {device}")

# -----
# 2. Simulated Historical Ticket Data (for fine-tuning)
# -----
# We've expanded the historical data slightly and added more variance.
historical_tickets = [
    {
        "summary": "My password for the corporate network is not working.",
        "category": "Password Reset",
        "resolution": "Initiated a password reset. User notified to set a new password."
    },
    {
        "summary": "I can't connect to the internet from my office PC.",
        "category": "Network Issue",
        "resolution": "Diagnosed and resolved a DNS configuration problem. Clearing browser cache."
    },
    {
        "summary": "Requesting installation of Adobe Photoshop on my work laptop.",
        "category": "Software Installation",
        "resolution": "Software installation request approved and completed. Adobe Creative Cloud installed."
    },
    {
        "summary": "My account is locked out, please help.",
        "category": "Account Lockout",
        "resolution": "Account lockout resolved. User has been sent an email with instructions to reset password."
    },
    {
        "summary": "My monitor is not turning on. I've checked the power cable."
    }
]

```

```

        "category": "Hardware Issue",
        "resolution": "Troubleshoot the monitor connection. Replaced the display p
    },
    {
        "summary": "I need to reset my password for my email account.",
        "category": "Password Reset",
        "resolution": "Successfully reset the user's email password. Provided a t
    },
    {
        "summary": "My keyboard is unresponsive, and some keys are not working.",
        "category": "Hardware Issue",
        "resolution": "The keyboard was found to be faulty. A new keyboard has be
    },
    {
        "summary": "I can't see the company's shared drive.",
        "category": "Network Issue",
        "resolution": "Re-mapped the network drive (H: drive) for the user. Verif
    },
    {
        "summary": "I need to install Microsoft Visio on my computer.",
        "category": "Software Installation",
        "resolution": "Software installation request for Microsoft Visio has beer
    }
]

```

```

# Convert the list of dictionaries to a pandas DataFrame
df = pd.DataFrame(historical_tickets)
display(Markdown("### Simulated Historical Ticket Data"))
display(df)

```

```

# -----
# 3. Data Preparation for Fine-Tuning a T5 Model
# -----
# T5 is a text-to-text model, so we format the input and output as strings.
# Input will be: "summarize: [ticket summary]"
# Output will be: "category: [category]. resolution: [resolution]"

```

```

model_name = "t5-small"
tokenizer = T5Tokenizer.from_pretrained(model_name)

```

```

def preprocess_data(examples):
    """
    Function to format the data for T5 fine-tuning.
    """
    inputs = [f"summarize: {summary}" for summary in examples['summary']]
    # The target text is a combination of the category and the resolution.
    targets = [f"category: {cat}. resolution: {res}" for cat, res in zip(examples
    )

    # Tokenize the inputs and targets
    model_inputs = tokenizer(inputs, max_length=512, truncation=True, padding="ma
    labels = tokenizer(targets, max_length=512, truncation=True, padding="max_ler

    # Set -100 for padding tokens in the labels so they are ignored in the loss c
    labels["input_ids"] = [

```

```

        [(1 if l != tokenizer.pad_token_id else -100) for l in label] for label i
    ]
    model_inputs['labels'] = labels['input_ids']

    return model_inputs

# Load the T5 model
print("\nLoading T5 model...")
model = T5ForConditionalGeneration.from_pretrained(model_name).to(device)
print("Model loaded.")

# Convert the pandas DataFrame to a Hugging Face Dataset
dataset = Dataset.from_pandas(df)
# Preprocess the dataset for the model
tokenized_dataset = dataset.map(preprocess_data, batched=True)

# -----
# 4. Fine-Tuning the Model
# -----
print("\nStarting fine-tuning...")

training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=100, # Increased epochs for better learning on a small data
    per_device_train_batch_size=2,
    warmup_steps=100,
    learning_rate=1e-4,
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=10,
    report_to="none"
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    tokenizer=tokenizer,
)

trainer.train()
print("\nFine-tuning complete.")

# -----
# 5. Testing the Fine-Tuned Model with the New Tickets
# -----
def process_new_ticket_finetuned(ticket_summary):
    """
    Processes a new ticket using the fine-tuned T5 model.
    """
    display(Markdown(f"### New Incoming Ticket:\n> '{ticket_summary}'"))

    # Prepare the input for the fine-tuned model

```

```

input_text = f"summarize: {ticket_summary}"
inputs = tokenizer(input_text, return_tensors="pt", max_length=512, truncatic

# Generate the output
outputs = model.generate(
    **inputs,
    max_length=200, # Increased max_length
    num_beams=4,
    early_stopping=True,
)

# Decode the output
generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)

# Debugging print to see the raw output
print(f"Raw generated text: {generated_text}")

# Extract category and resolution using a more robust regex
category = "Unknown"
resolution = "N/A"

match = re.search(r"category: (.*)\. resolution: (.*)", generated_text)
if match:
    category = match.group(1).strip()
    resolution = match.group(2).strip()
else:
    # Fallback if regex fails
    resolution = generated_text


# Present the final result
display(Markdown("---"))
display(Markdown("### IntelliAssist's Analysis and Response (Fine-Tuned Model"))
display(Markdown(f"***Predicted Category:** `{category}`"))
display(Markdown(f"***Draft Resolution:**\n> {resolution}"))
display(Markdown("---"))

if __name__ == "__main__":
    # The challenging ticket that caused an issue before
    new_ticket_to_test = "My mouse is not working, and I can't click on anything.
    process_new_ticket_finetuned(new_ticket_to_test)

    # Another new ticket to see how it performs
    new_ticket_6 = "I am unable to see the company's shared drive on my desktop."
    process_new_ticket_finetuned(new_ticket_6)

    # A new password reset ticket to verify it still works
    new_ticket_7 = "I need to change my password because it expired."
    process_new_ticket_finetuned(new_ticket_7)

```

 Libraries installed and imported successfully.
Using device: cuda

Simulated Historical Ticket Data

	summary	category	resolution
0	My password for the corporate network is not w...	Password Reset	Initiated a password reset. User notified to s...
1	I can't connect to the internet from my office...	Network Issue	Diagnosed and resolved a DNS configuration pro...
2	Requesting installation of Adobe Photoshop on ...	Software Installation	Software installation request approved and com...
3	My account is locked out, please help.	Account Lockout	Account lockout resolved. User has been sent a...
4	My monitor is not turning on. I've checked the...	Hardware Issue	Troubleshooted the monitor connection. Replaced t...
5	I need to reset my password for my email account.	Password Reset	Successfully reset the user's email password. ...
6	My keyboard is unresponsive, and some keys are...	Hardware Issue	The keyboard was found to be faulty. A new key...
7	I can't see the company's shared drive.	Network Issue	Re-mapped the network drive (H: drive) for the...
8	I need to install Microsoft Visio on my computer.	Software Installation	Software installation request for Microsoft Vi...

/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>)
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.
warnings.warn(
You are using the default legacy behaviour of the <class 'transformers.models.t5.token

Loading T5 model...
Model loaded.

Map: 100% 9/9 [00:00<00:00, 46.33 examples/s]

Starting fine-tuning...
/tmp/ipython-input-2363760717.py:144: FutureWarning: `tokenizer` is deprecated and will be removed in a future version. Use `processing_class` instead.
trainer = Trainer(
[500/500 01:31, Epoch 100/100]

Step	Training Loss
10	5.098000
20	5.021000
30	4.787000
40	4.437600
50	4.155200

60	3.822500
70	3.566400
80	3.247800
90	2.781100
100	2.366700
110	2.049800
120	1.706800
130	1.440300
140	1.205900
150	1.027500
160	0.926000
170	0.830800
180	0.763300
190	0.711300
200	0.617200
210	0.518000
220	0.489700
230	0.486300
240	0.364500
250	0.344900
260	0.329000
270	0.297600
280	0.319100
290	0.267800
300	0.241400
310	0.243400
320	0.217300
330	0.177700
340	0.180400
350	0.156800
360	0.159400
370	0.186400
380	0.150600
390	0.147100

400	0.122200
410	0.135400
420	0.110900
430	0.101700
440	0.105100
450	0.122500
460	0.101900
470	0.106300
480	0.101700
490	0.111600
500	0.094800

Fine-tuning complete.

New Incoming Ticket:

'My mouse is not working, and I can't click on anything.'

Raw generated text: category: The mouse was found to be ok. resolution: The mouse was

IntelliAssist's Analysis and Response (Fine-Tuned Model)

Predicted Category: The mouse was found to be ok

Draft Resolution:

The mouse was found to be faulty. A new mouse has been assigned to the user.

New Incoming Ticket:

'I am unable to see the company's shared drive on my desktop.'

Raw generated text: category: Hardware Issue. resolution: Diagnosed and managed to fi

IntelliAssist's Analysis and Response (Fine-Tuned Model)

Predicted Category: Hardware Issue

Draft Resolution:

Diagnosed and managed to find a new drive on the user's desktop.

New Incoming Ticket:

'I need to change my password because it expired.'

Raw generated text: category: Password Issued. resolution: Short-listed and resolved

IntelliAssist's Analysis and Response (Fine-Tuned Model)

Predicted Category: Password Issued

Draft Resolution:

Short-listed and resolved to change. Provided a new password and instructions to change it.

```
# A new data point to test the fine-tuned model's ability to generalize
new_ticket_8 = "I need to request a new user account for our new employee."
process_new_ticket_finetuned(new_ticket_8)
```

**New Incoming Ticket:**

'I need to request a new user account for our new employee.'

Raw generated text: category: User Account. resolution: User's request for a new user

IntelliAssist's Analysis and Response (Fine-Tuned Model)

Predicted Category: User Account

Draft Resolution:

User's request for a new user account has been approved and approved.
