#from transformers import BertTokenizer, BertForSequenceClassification, Trainer, TrainingArguments

#from sklearn.model_selection import train_test_split

#from sklearn.metrics import accuracy_score, precision_recall_fscore_support ! pip install pyarrow

Step 01: Import all the necessary libraries

```
import os
import warnings
import numpy as np
import pandas as pd

# from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

from datasets import Dataset, DatasetDict
from datasets.features import Value, ClassLabel
from datasets import Features



from transformers import DistilBertTokenizerFast
from transformers import DistilBertTokenizer,
DistilBertForSequenceClassification

from transformers import TrainingArguments
from transformers import Trainer
from transformers import DataCollatorWithPadding

# from torch.utils.data import DataLoader, TensorDataset, RandomSampler,
SequentialSampler
import torch


/home/bob35/GBC/VScode4Pro/.venv/lib/python3.11/site-
packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
2024-09-29 00:35:30.392184: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unable to
register cuFFT factory: Attempting to register factory for plugin cuFFT when
one has already been registered
2024-09-29 00:35:30.405919: E
```

```
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Unable to
register cuDNN factory: Attempting to register factory for plugin cuDNN when
one has already been registered
2024-09-29 00:35:30.410017: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Unable to
register cuBLAS factory: Attempting to register factory for plugin cuBLAS
when one has already been registered
2024-09-29 00:35:30.420888: I
tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is
optimized to use available CPU instructions in performance-critical
operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
2024-09-29 00:35:31.247713: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could
not find TensorRT
```

```
# Suppress HF_HUB_DISABLE_SYMLINKS_WARNING
# os.environ['HF_HUB_DISABLE_SYMLINKS_WARNING'] = '1'

# Suppress FutureWarning from transformers
# warnings.filterwarnings('ignore', category=FutureWarning,
module='transformers')

# Cuda can be applied to accelerate computing if cuda.is_available()
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f'Using device: {device}')
os.environ['CUDA_LAUNCH_BLOCKING'] = '1'
os.environ['TF_FORCE_GPU_ALLOW_GROWTH']='true'
os.environ["PYTORCH_CUDA_ALLOC_CONF"] = "max_split_size_mb:512"
torch.cuda.empty_cache()
# import gc
# del variables
# gc.collect()
```

```
Using device: cuda
```

Step 02: Data Preparation

```
# Import CSV
# primarily use the Query and Product columns.
# path = r'C:\Users\data' # or whatever your path

df = pd.read_csv('data/customer_queries.csv',engine="pyarrow")
# df = pd.read_csv('data/customer_queries.csv')

# Take a look at the dataframe
print('Shape of the data= ', df.shape)
print('Column datatypes= \n',df.dtypes)
print(df.columns)
df.describe()
```

```
Shape of the data=  (10000, 4)
Column datatypes=
 Date       object
Product    object
Query      object
Answer     object
dtype: object
Index(['Date', 'Product', 'Query', 'Answer'], dtype='object')
```

	Date	Product	Query	Answer
count	10000	10000	10000	10000
unique	84	20	40	40
top	1/1/2022	Printer	How to connect the Printer to a computer?	To connect the Printer to a computer, use the ...
freq	120	500	250	250

```
label2idx = {label:i for i, label in enumerate(df.Product.unique().tolist())}
label2idx
```

```
{'Printer': 0,
 'Scanner': 1,
 'Laptop': 2,
 'Monitor': 3,
 'Keyboard': 4,
 'Mouse': 5,
 'Projector': 6,
 'Fax machine': 7,
 'Calculator': 8,
 'Shredder': 9,
 'Photocopier': 10,
 'Whiteboard': 11,
 'Paper shredder': 12,
 'Desk lamp': 13,
 'External hard drive': 14,
 'Conference phone': 15,
 'Label maker': 16,
 'Document camera': 17,
 'Wireless presenter': 18,
 'USB hub': 19}
```

```
# Reverse label map
idx2label = {v:k for k,v in label2idx.items()}
idx2label
```

```
{0: 'Printer',
 1: 'Scanner',
```

```
 2: 'Laptop',
 3: 'Monitor',
 4: 'Keyboard',
 5: 'Mouse',
 6: 'Projector',
 7: 'Fax machine',
 8: 'Calculator',
 9: 'Shredder',
 10: 'Photocopier',
 11: 'Whiteboard',
 12: 'Paper shredder',
 13: 'Desk lamp',
 14: 'External hard drive',
 15: 'Conference phone',
 16: 'Label maker',
 17: 'Document camera',
 18: 'Wireless presenter',
 19: 'USB hub'}
```

```
# Map Categories to labels
# assign numeric labels to each category
# Create a new column with integer mapping to classes.
df['label'] = df.Product.map(label2idx)
```

```
df
```

```
            Date             Product  \
0       1/1/2022             Printer
1       2/2/2022             Scanner
2       3/3/2022              Laptop
3       4/4/2022             Monitor
4       5/5/2022            Keyboard
...          ...                 ...
9995  12/28/2022    Conference phone
9996    1/1/2022         Label maker
9997    2/2/2022     Document camera
9998    3/3/2022  Wireless presenter
9999    4/4/2022             USB hub

                                                  Query  \
0             How to connect the Printer to a computer?
1               What are the dimensions of the Scanner?
2     Can the Laptop be used with both Windows and M...
3     How to replace the ink cartridges in the Monitor?
4       What is the maximum resolution of the Keyboard?
...                                                 ...
9995  What is the maximum zoom level of the Conferen...
9996        Is the Label maker compatible with USB 3.0?
9997   How to enable sleep mode on the Document camera?
9998      What is the weight of the Wireless presenter?
```

```
9999                    Can the USB hub be wall-mounted?

                                                 Answer  label
0     To connect the Printer to a computer, use the ...      0
1     The dimensions of the Scanner are 10 x 8 x 5 i...      1
2     Yes, the Laptop is compatible with both Window...      2
3     To replace the ink cartridges in the Monitor, ...      3
4     The maximum resolution of the Keyboard is 1200...      4
...                                                 ...    ...
9995  The maximum zoom level of the Conference phone...     15
9996   Yes, the Label maker is compatible with USB 3.0.     16
9997  To enable sleep mode on the Document camera, a...     17
9998  The weight of the Wireless presenter is 3.5 po...     18
9999  Yes, the USB hub can be wall-mounted. Use the ...     19

[10000 rows x 5 columns]
```

```
df.info()
```

```
df.drop(columns=['Date', 'Product','Answer'], inplace=True)
```

```
# Reset index
df.reset_index(drop=True, inplace=True)
```

```
df
```

```
# Split the data into train, test and validation sets
# shuffling the  data so that no order bias is there
```

```
train_df, test_df = train_test_split(df, test_size=0.2, shuffle=True,
stratify=df['label'], random_state=42)
train_df, eval_df = train_test_split(train_df, test_size=0.2, shuffle=True,
stratify=train_df['label'], random_state=42)
```

```
# Encode labels
# label_encoder = LabelEncoder()
# df['Product'] = label_encoder.fit_transform(df['Product'])
train_df
# Split data
# X_train, X_test, y_train, y_test = train_test_split(df['Query'],
df['Product'], test_size=0.2, random_state=42)
```

```
                                                  Query  label
3808  What are the available connectivity options fo...      8
4676        Is the Label maker compatible with USB 3.0?     16
1055  What is the maximum scanning speed of the Conf...     15
1965    Is the Mouse compatible with wireless printing?      5
5887  How to calibrate the color settings on the Fax...      7
...                                                 ...    ...
6074  How to replace the toner cartridge in the Exte...     14
2432    What is the display size of the Paper shredder?     12
```

```
7643  How to replace the ink cartridges in the Monitor?      3
2923  How to replace the ink cartridges in the Monitor?      3
3132  How to troubleshoot common issues with the Pap...     12

[6400 rows x 2 columns]
```

```
train_df.shape, test_df.shape, eval_df.shape
```

```
((6400, 2), (2000, 2), (1600, 2))
```

```
test_df.label.value_counts(normalize=True)
```

```
label
4     0.05
0     0.05
8     0.05
18    0.05
5     0.05
19    0.05
12    0.05
6     0.05
13    0.05
10    0.05
16    0.05
9     0.05
2     0.05
15    0.05
14    0.05
7     0.05
11    0.05
17    0.05
3     0.05
1     0.05
Name: proportion, dtype: float64
```

```
eval_df.label.value_counts(normalize=True)
```

Step 03: Create Features

```
# two features, first is "Query" which is type string and the other is our
target "label".

features=Features({"Query": Value(dtype='string', id=None),
                "label": ClassLabel(num_classes=20,

names=['Printer','Scanner','Laptop','Monitor','Keyboard','Mouse','Projector',
'Fax machine','Calculator','Shredder','Photocopier','Whiteboard','Paper
shredder','Desk lamp','External hard drive','Conference phone','Label
maker','Document camera','Wireless presenter','USB hub'], id=None)})
```

```
# Use from pandas to direclty convert from pandas dataframe to dataset.
train_dataset = Dataset.from_pandas(train_df,
features=features,preserve_index=False)
test_dataset = Dataset.from_pandas(test_df,
features=features,preserve_index=False)
eval_dataset = Dataset.from_pandas(eval_df,
features=features,preserve_index=False)

#  Create a dataset dict combining all the datasets under one.
dataset = DatasetDict({"train": train_dataset, "test": test_dataset,
"validation": eval_dataset})
```

Step 04: Tokenization Use the BERT tokenizer to convert text into tokens that BERT can understand. This includes adding special tokens, padding, and truncating to a fixed length. Use DistilBert, a derivative of BERT model.

```
# Load tokenizer from huggingface
# Initialize BERT tokenizer
# tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
# model_checkpoint = "distilbert-base-uncased"
# tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
# from transformers import DataCollatorWithPadding

def tokenize_function(examples):
    return tokenizer(examples["Query"], truncation=True,
padding="max_length", max_length=256)

tokenizer = DistilBertTokenizerFast.from_pretrained("distilbert-base-
uncased")
# tokenized_datasets = dataset.map(tokenize_function, batched=True,
batch_size=1000)
tokenized_datasets = dataset.map(tokenize_function, batched=True,
batch_size=100)

# Dynamic Padding
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

```
/home/bob35/GBC/VScode4Pro/.venv/lib/python3.11/site-
packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by
default. This behavior will be depracted in transformers v4.45, and will be
then set to `False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
Map: 100%|██████████| 6400/6400 [00:00<00:00, 14238.95 examples/s]
Map: 100%|██████████| 2000/2000 [00:00<00:00, 12756.69 examples/s]
Map: 100%|██████████| 1600/1600 [00:00<00:00, 13391.03 examples/s]
```

Step 05: Load the Model initialize our model with num_labels=20 as we have 20 classes to predict

```
model = DistilBertForSequenceClassification.from_pretrained("distilbert-base-uncased", num_labels=20)
model.to(device)
```

```
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncased and are newly initialized: ['classifier.bias', 'classifier.weight', 'pre_classifier.bias', 'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
DistilBertForSequenceClassification(
  (distilbert): DistilBertModel(
    (embeddings): Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (transformer): Transformer(
      (layer): ModuleList(
        (0-5): 6 x TransformerBlock(
          (attention): MultiHeadSelfAttention(
            (dropout): Dropout(p=0.1, inplace=False)
            (q_lin): Linear(in_features=768, out_features=768, bias=True)
            (k_lin): Linear(in_features=768, out_features=768, bias=True)
            (v_lin): Linear(in_features=768, out_features=768, bias=True)
            (out_lin): Linear(in_features=768, out_features=768, bias=True)
          )
          (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (ffn): FFN(
            (dropout): Dropout(p=0.1, inplace=False)
            (lin1): Linear(in_features=768, out_features=3072, bias=True)
            (lin2): Linear(in_features=3072, out_features=768, bias=True)
            (activation): GELUActivation()
          )
          (output_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        )
      )
    )
  )
  (pre_classifier): Linear(in_features=768, out_features=768, bias=True)
  (classifier): Linear(in_features=768, out_features=20, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

Step 06: Training Define TrainingArugments

```
'''
    # Number of times the entire training dataset will pass through the model
    num_train_epochs=3,  # Train the model for 3 epochs

    # Batch size per device (GPU/CPU) during training
    per_device_train_batch_size=16,  # 16 training samples per batch per
device

    # Batch size per device during evaluation
    per_device_eval_batch_size=64,  # 64 evaluation samples per batch per
device

    # Number of warmup steps for the learning rate scheduler (helps with
better convergence)
    warmup_steps=500,  # Gradually increase learning rate during first 500
steps

    # Weight decay for regularization to prevent overfitting (penalizes large
weights)
    weight_decay=0.01,  # 1% weight decay
'''

training_args = TrainingArguments(output_dir="./results",
                                  per_device_train_batch_size=32,
                                  per_device_eval_batch_size=16,
                                  learning_rate=5e-5,
                                  num_train_epochs=5,
                                  optim='adamw_torch_fused',
                                  logging_dir="./logs/",
                                  logging_strategy='steps',
                                  logging_steps=200,
                                  evaluation_strategy='epoch',
                                  save_strategy='epoch',
                                  save_total_limit=2,
                                  load_best_model_at_end=True)
```

```
/home/bob35/GBC/VScode4Pro/.venv/lib/python3.11/site-
packages/transformers/training_args.py:1525: FutureWarning:
`evaluation_strategy` is deprecated and will be removed in version 4.46 of
🤗 Transformers. Use `eval_strategy` instead
  warnings.warn(
```

```
# Initialize the Trainer class
trainer = Trainer(model=model,
                  args=training_args,
                  train_dataset=tokenized_datasets['train'],
                  eval_dataset=tokenized_datasets['validation'],
                  data_collator=data_collator,
                  tokenizer=tokenizer)
```

```
%%time
trainer.train()
```

```
 20%|██        | 200/1000 [01:23<05:36,  2.38it/s]
```

```
{'loss': 0.5507, 'grad_norm': 0.0975455790758133, 'learning_rate': 4e-05,
'epoch': 1.0}
```

```
 20%|██        | 200/1000 [01:31<05:36,  2.38it/s]
```

```
{'eval_loss': 0.011260993778705597, 'eval_runtime': 7.8272,
'eval_samples_per_second': 204.414, 'eval_steps_per_second': 12.776, 'epoch':
1.0}
```

```
 40%|████      | 400/1000 [02:57<04:14,  2.36it/s]
```

```
{'loss': 0.0096, 'grad_norm': 0.036252655088901 52, 'learning_rate': 3e-05,
'epoch': 2.0}
```

```
 40%|████      | 400/1000 [03:05<04:14,  2.36it/s]
```

```
{'eval_loss': 0.003672544378787279, 'eval_runtime': 7.8901,
'eval_samples_per_second': 202.786, 'eval_steps_per_second': 12.674, 'epoch':
2.0}
```

```
 60%|██████    | 600/1000 [04:31<02:50,  2.35it/s]
```

```
{'loss': 0.0044, 'grad_norm': 0.02112532965838909, 'learning_rate': 2e-05,
'epoch': 3.0}
```

```
 60%|██████    | 600/1000 [04:39<02:50,  2.35it/s]
```

```
{'eval_loss': 0.0021333941258490086, 'eval_runtime': 7.9475,
'eval_samples_per_second': 201.322, 'eval_steps_per_second': 12.583, 'epoch':
3.0}
```

```
 80%|████████  | 800/1000 [06:06<01:25,  2.35it/s]
```

```
{'loss': 0.003, 'grad_norm': 0.02154717780649662, 'learning_rate': 1e-05,
'epoch': 4.0}
```

```
 80%|████████  | 800/1000 [06:14<01:25,  2.35it/s]
```

```
{'eval_loss': 0.0016024267533794045, 'eval_runtime': 7.9481,
'eval_samples_per_second': 201.307, 'eval_steps_per_second': 12.582, 'epoch':
4.0}
```

```
100%|██████████| 1000/1000 [07:40<00:00,  2.35it/s]
```

```
{'loss': 0.0025, 'grad_norm': 0.015545379370450974, 'learning_rate': 0.0,
'epoch': 5.0}


100%|██████████| 1000/1000 [07:49<00:00,  2.35it/s]

{'eval_loss': 0.0014572575455531478, 'eval_runtime': 8.1446,
'eval_samples_per_second': 196.45, 'eval_steps_per_second': 12.278, 'epoch':
5.0}

100%|██████████| 1000/1000 [07:50<00:00,  2.12it/s]

{'train_runtime': 470.6819, 'train_samples_per_second': 67.986,
'train_steps_per_second': 2.125, 'train_loss': 0.11403252986073494, 'epoch':
5.0}
CPU times: user 7min 47s, sys: 3.56 s, total: 7min 51s
Wall time: 7min 51s



TrainOutput(global_step=1000, training_loss=0.11403252986073494,
metrics={'train_runtime': 470.6819, 'train_samples_per_second': 67.986,
'train_steps_per_second': 2.125, 'total_flos': 2120158740480000.0,
'train_loss': 0.11403252986073494, 'epoch': 5.0})

# Save the model's state dictionary (recommended method):
torch.save(model.state_dict(), './results/model_distill_bert.pth')
```

Step 07: Evaluation Evaluate the model on the test set:

```
eval_results = trainer.evaluate()
print(f"Evaluation results: {eval_results}")

100%|██████████| 100/100 [00:07<00:00, 13.24it/s]

Evaluation results: {'eval_loss': 0.0014572575455531478, 'eval_runtime':
7.6587, 'eval_samples_per_second': 208.914, 'eval_steps_per_second': 13.057,
'epoch': 5.0}
```

Step 08: Inference Use the trained model to make predictions on new data:

```
# Load the saved state dict
# model.load_state_dict(torch.load('./results/model_distill_bert.pth'))

# Move the model to the appropriate device
# model = model.to(device)

# Set the model to evaluation mode
model.eval()
```

```
# Example usage
# Inference
query = "How do I connect my printer to Wi-Fi?"
inputs = tokenizer(query, return_tensors="pt")

# Move input tensors to the same device as the model
inputs = {k: v.to(device) for k, v in inputs.items()}

with torch.no_grad():
    outputs = model(**inputs)

prediction = torch.argmax(outputs.logits).item()

print(f'Predicted Product: {prediction}')

Predicted Product: 0
```

Step 09: Build a Inference pipeline

```
from transformers import DistilBertTokenizer,
DistilBertForSequenceClassification, pipeline
# Define the feature list
feature_list = {
    0: 'Printer',
    1: 'Scanner',
    2: 'Laptop',
    3: 'Monitor',
    4: 'Keyboard',
    5: 'Mouse',
    6: 'Projector',
    7: 'Fax machine',
    8: 'Calculator',
    9: 'Shredder',
    10: 'Photocopier',
    11: 'Whiteboard',
    12: 'Paper shredder',
    13: 'Desk lamp',
    14: 'External hard drive',
    15: 'Conference phone',
    16: 'Label maker',
    17: 'Document camera',
    18: 'Wireless presenter',
    19: 'USB hub'
}


def load_saved_model(model_path, device="cuda" if torch.cuda.is_available()
else "cpu"):
    """
    Load the saved model and tokenizer
```

```
    """
    tokenizer = DistilBertTokenizerFast.from_pretrained("distilbert-base-
uncased")
    # model =
DistilBertForSequenceClassification.from_pretrained(model_path).to(device)
    model.eval()
    return tokenizer, model

# tokenizer, model = load_saved_model('./results/model_distill_bert.pth')
# Create a classification pipeline
model.eval()
classification_pipeline = pipeline('text-classification', model=model,
tokenizer=tokenizer, device=device)



# Sample texts to test
sample_questions = [
    "Hello, how are you?",
    "What is the maximum scanning speed of the Conference phone?",
    "How to set up the Document camera for wireless networking?",
    " How to conect Laptop to Printer "
    # Add more sample texts as needed
]

# Generate and print outputs
# Classify each question
for question in sample_questions:
    result = classification_pipeline(question)
    label_id = int(result[0]['label'].split('_')[-1])
    feature = feature_list.get(label_id, "Unknown")
    print(f"Question: {question}\nClassified as: {feature}\n")
```

```
Question: Hello, how are you?
Classified as: Scanner

Question: What is the maximum scanning speed of the Conference phone?
Classified as: Conference phone

Question: How to set up the Document camera for wireless networking?
Classified as: Document camera

Question:  How to conect Laptop to Printer
Classified as: Printer
```