BERTによるファインチューニング

ゴール

- 事前学習済みのBERTモデルを用いて映画レビューの **ポジティブ/ネガティブ分類**
- 学習済みモデルでテキスト分類の推論ができるようになる

データセットの準備

```
from datasets import load_dataset

# IMDBレビューのデータセットをロード
dataset = load_dataset("imdb")

# サンプル表示
print(dataset)
print(dataset["train"][0])
```

Tokenizerで前処理

```
from transformers import BertTokenizer

# 事前学習済みのBERTトークナイザー
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# データセットにトークナイザーを適用
def tokenize_function(examples):
    return tokenizer(
        examples["text"],
        padding="max_length",
        truncation=True,
        max_length=128
    )

# 全データに適用
tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

データセット分割

```
# 不要なカラムを削除
tokenized_datasets = tokenized_datasets.remove_columns(["text"])
tokenized_datasets = tokenized_datasets.rename_column("label", "labels")
tokenized_datasets.set_format("torch")  # PyTorchテンソル化

# 分割
#train data
train_dataset = tokenized_datasets["train"].shuffle(seed=42).select(range(5000))

#test data
eval_dataset = tokenized_datasets["test"].shuffle(seed=42).select(range(1000))
```

モデル定義

```
from transformers import BertForSequenceClassification

# BERTを二値分類用にロード
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=2)
```

Trainerによる学習設定

```
from transformers import TrainingArguments, Trainer

# 学習設定
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=2,       # エポック数
    learning_rate=2e-5,         #学習率
    weight_decay=0.01,          #正則化
)

# Trainerのセットアップ
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset
)
```

学習とモデル評価

```
# 学習
trainer.train()

# 評価
metrics = trainer.evaluate()
print(metrics)
```

推論

```
# 推論用テキスト
test_text = "The movie was fantastic! I loved every moment."

# トークナイズ
inputs = tokenizer(test_text, return_tensors="pt", padding=True, truncation=True, max_length=128)

# モデル推論
outputs = model(**inputs)
predicted_class = outputs.logits.argmax().item()

# 結果表示
if predicted_class == 1:
    print("予測: ポジティブレビュー 👍")
else:
    print("予測: ネガティブレビュー 👎")
```