perplexity

各種意圖預測方法的 Python 實作範例

以下針對先前提出的六大類方法，每種各提供一段示範性 Python 程式碼。程式中以常見開源工具（如 Hugging Face Transformers、SentenceTransformers、Faiss、PyTorch、scikit-learn 等）為例，示範核心流程。實務中需依據自有資料與需求做擴充與調校。

1. 基於大語言模型的意圖識別方法

1.1 LLM 驅動的意圖分類（Fine-tune BERT）

```
from transformers import BertTokenizerFast, BertForSequenceClassification, Trainer, Trair
import torch
from datasets import load_dataset

# 1. 載入數據集（範例：CSV 包含 text, intent_label）
ds = load_dataset("csv", data_files="intent_data.csv")
tokenizer = BertTokenizerFast.from_pretrained("bert-base-chinese")

def preprocess(examples):
    return tokenizer(examples["text"], truncation=True, padding=True)
ds = ds.map(preprocess, batched=True)
ds = ds.rename_column("intent_label", "labels")
ds.set_format(type="torch", columns=["input_ids","attention_mask","labels"])

# 2. 載入預訓練模型
model = BertForSequenceClassification.from_pretrained("bert-base-chinese", num_labels=ds[

# 3. 訓練設定
args = TrainingArguments(
    output_dir="out_llm_intent",
    evaluation_strategy="epoch",
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    logging_dir="logs"
)
trainer = Trainer(model=model, args=args, train_dataset=ds["train"], eval_dataset=ds["val
trainer.train()
```

1.2 語義路由（Semantic Routing with Sentence-Transformers + Faiss）

```
from sentence_transformers import SentenceTransformer
import faiss
import numpy as np

# 1. 定義各意圖的示例句
```

```
intents = {
    "查天氣": ["今天的天氣如何？","明天會下雨嗎？"],
    "設定鬧鐘": ["幫我設個鬧鐘","早上七點叫我起床"]
}
texts = sum(intents.values(), [])
labels = sum([[k]*len(v) for k,v in intents.items()], [])

# 2. 建嵌入與索引
model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")
embeddings = model.encode(texts, convert_to_numpy=True)
index = faiss.IndexFlatL2(embeddings.shape[^1])
index.add(embeddings)

# 3. 查詢路由
def route(query):
    q_emb = model.encode([query])
    D,I = index.search(q_emb, k=2)
    # 基於最近鄰投票
    neigh_labels = [labels[i] for i in I[^0]]
    return max(set(neigh_labels), key=neigh_labels.count)

print(route("幫我明天早上叫醒我"))  # -> 設定鬧鐘
```

2. 檢索增強生成（RAG）協作方法

2.1 REIC：RAG 增強意圖分類（向量檢索 + LLM 分類）

```
from sentence_transformers import SentenceTransformer
from openai import OpenAI

# 1. 建立 (query, intent) 向量索引
kb = [("查天氣","查天氣"),("設鬧鐘","設定鬧鐘")]
model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")
kb_emb = model.encode([q for q,_ in kb], convert_to_numpy=True)

# 2. Faiss 建索引
import faiss
index = faiss.IndexFlatL2(kb_emb.shape[^1])
index.add(kb_emb)

# 3. 檢索 + LLM 驗證
openai = OpenAI()
def classify_rag(query):
    qe = model.encode([query])
    D,I = index.search(qe, k=3)
    candidates = [kb[i][^1] for i in I[^0]]
    prompt = f"以下三個意圖：{candidates}。請判斷最適合查詢“{query}”的意圖。"
    resp = openai.chat.completions.create(model="gpt-4o", messages=[{"role":"user","conte
    return resp.choices[^0].message.content

print(classify_rag("今天上海氣溫"))  # -> 查天氣
```

3. 查詢理解與擴展技術

3.1 RQ-RAG：查詢重寫（使用 LLM）

```
from openai import OpenAI

client = OpenAI()
def rewrite(query):
    system = "你是查詢重寫專家，請將用戶查詢轉換為更精準的檢索語句。"
    resp = client.chat.completions.create(
        model="gpt-4o",
        messages=[
            {"role":"system","content":system},
            {"role":"user","content":query}
        ]
    )
    return resp.choices[^0].message.content

print(rewrite("明天會怎樣？"))  # -> 重寫為：明天北京的天氣預報
```

3.2 多查詢生成（Query Expansion）

```
from transformers import pipeline

expander = pipeline("text2text-generation", model="facebook/bart-large")
def expand(query):
    prompt = f"請基於 '{query}' 生成三個相關的檢索查詢："
    return expander(prompt, max_length=64)[^0]["generated_text"]

print(expand("股票價格"))  # -> 生成多個同義或細分類查詢
```

4. 用戶偏好建模與適應性方法

4.1 個性化意圖識別（行為特徵 + LLM）

```
import numpy as np
from sklearn.linear_model import LogisticRegression
# 假設有用戶過往行為特徵 user_feats 和查詢 query_emb
user_feats = np.random.rand(100, 8)  # 用戶向量
query_emb = np.random.rand(100, 256)  # 查詢 embedding
X = np.concatenate([user_feats, query_emb], axis=1)
y = np.random.randint(0, 3, size=100)
clf = LogisticRegression().fit(X, y)

def personalized_intent(user_vec, query_vec):
    feat = np.concatenate([user_vec, query_vec])
    return clf.predict([feat])[^0]
```

4.2 RAGate：動態檢索門控

```
def ragate(query, context_history):
    # 信心度低時呼叫 RAG
    confidence = llm_confidence(query, context_history)
    if confidence < 0.7:
        return classify_rag(query)
    else:
        return canned_response(query)

print(ragate("今天股票怎樣？", []))
```

5. 記憶增強與上下文建模

5.1 MANNs：記憶增強神經網絡（PyTorch 範例）

```
import torch, torch.nn as nn

class MANNCell(nn.Module):
    def __init__(self, input_dim, mem_dim):
        super().__init__()
        self.write = nn.Linear(input_dim + mem_dim, mem_dim)
        self.read = nn.Linear(input_dim + mem_dim, mem_dim)
    def forward(self, x, mem):
        concat = torch.cat([x, mem], dim=-1)
        new_mem = torch.tanh(self.write(concat))
        read_vec = torch.tanh(self.read(concat))
        return read_vec, new_mem

# 在 RNN 中使用 MANNCell
```

5.2 對話上下文建模（Transformer + History）

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

def chat_response(history, query):
    prompt = "\n".join(history + [f"User: {query}", "Bot:"])
    inputs = tokenizer(prompt, return_tensors="pt")
    out = model.generate(**inputs, max_length=inputs.input_ids.shape[^1]+50)
    reply = tokenizer.decode(out[^0][inputs.input_ids.shape[^1]:], skip_special_tokens=Tr
    history.append(f"User: {query}")
    history.append(f"Bot: {reply}")
    return reply

hist = []
print(chat_response(hist, "你好"))
```

6. 少樣本學習與對比學習

6.1 對比學習意圖檢測（SimCLR 風格）

```
import torch.nn.functional as F

def contrastive_loss(z_i, z_j, temperature=0.5):
    z = torch.cat([z_i, z_j], dim=0)
    sim = F.cosine_similarity(z.unsqueeze(1), z.unsqueeze(0), dim=-1)
    sim_exp = torch.exp(sim / temperature)
    mask = (~torch.eye(2*z_i.size(0), dtype=bool)).float()
    return -torch.log(sim_exp * mask / (sim_exp.sum(dim=1, keepdim=True)*mask)).mean()
```

6.2 開放意圖檢測（基於閾值的 OOD）

```
def detect_out_of_scope(query_emb, class_centroids, threshold=0.6):
    sims = F.cosine_similarity(query_emb, class_centroids)
    if sims.max() < threshold:
        return "未知意圖"
    else:
        return f"意圖_{sims.argmax().item()}"
```

以上範例示範各類方法的**核心程式框架**，實務上仍須結合具體數據集、超參數調校及效能評估，才能真正落地應用。

⁂