TypeProfile: User data recovery

Overview

- You ask the user to type a fixed phrase (e.g., `.tie5Roanl`) in a local app.
- You use `pynput` to capture keypresses with timestamps.
- Extract 31 features: dwell times, flight times, and statistical summaries.
- Feed the feature vector to your trained ML model.
- Use an LLM (via API or offline model) to generate advice (e.g., "You press too hard on vowels", "Typing rhythm is inconsistent").

🔧 Step-by-Step Implementation (Local Python Version)

✅ Step 1: Install the Required Packages

```
bash
CopyEdit
pip install pynput numpy
```

✅ Step 2: Python Code to Capture Dwell and Flight Time

```
python
CopyEdit
from pynput import keyboard
import time
import numpy as np
```

```
typed_keys = []
timestamps = []

# Track key presses and releases
def on_press(key):
    try:
        typed_keys.append((key.char, 'down', time.time()))
    except AttributeError:
        pass

def on_release(key):
    try:
        typed_keys.append((key.char, 'up', time.time()))
        if key.char == 'l':  # Last letter of the fixed phrase `.tie5Roanl`
            return False
    except AttributeError:
        pass

# Start listener
print("Type the phrase: .tie5Roanl")
with keyboard.Listener(on_press=on_press, on_release=on_release) as listene
r:
    listener.join()
```

✅ Step 3: Process Features (Dwell, Flight, Stats)

```
python
CopyEdit
# Group press and release times
key_down_times = {}
dwell_times = []
flight_times = []

for i in range(len(typed_keys)):
```

```
    key, event_type, t = typed_keys[i]
    if event_type == 'down':
        key_down_times[key] = t
        if i > 0 and typed_keys[i-1][1] == 'up':
            flight_times.append(t - typed_keys[i-1][2])  # Flight time = current dow
n - previous up
    elif event_type == 'up' and key in key_down_times:
        dwell = t - key_down_times[key]
        dwell_times.append(dwell)

# Feature vector
feature_vector = []
feature_vector.extend(dwell_times[:10])     # First 10 dwell times
feature_vector.extend(flight_times[:10])    # First 10 flight times
feature_vector.append(sum(dwell_times))     # Total dwell time
feature_vector.append(sum(flight_times))    # Total flight time
feature_vector.append(np.mean(dwell_times))
feature_vector.append(np.mean(flight_times))
feature_vector.append(np.std(dwell_times))
feature_vector.append(np.std(flight_times))
feature_vector.append(typed_keys[-1][2] - typed_keys[0][2])  # Total typing d
uration

# Pad if needed
while len(feature_vector) < 31:
    feature_vector.append(0.0)

print("\nExtracted 31 Feature Vector:\n", np.round(feature_vector, 4))
```

📤 Step 4: Use It with Your Model

You can now feed `feature_vector` into your **XGBoost** or **RandomForestClassifier**, then pass the prediction to an LLM for advice like:

```
python
CopyEdit
typing_profile = model.predict([feature_vector])[0]

# Optionally use GPT or local LLM to generate feedback:
prompt = f"My typing profile was classified as '{typing_profile}'. How can I im
prove it?"
response = your_llm.generate(prompt)  # Can be OpenAI, LLaMA, Mistral etc.
```