model_evaluation_full_output

December 17, 2025

[1]:
```python
import requests
import time
import pandas as pd
from pathlib import Path
import os
import sys
import json
import ast
from datetime import datetime
from dotenv import load_dotenv

# Fix Path to import src
backend_path = Path("../teledoc-backend").resolve()
if str(backend_path) not in sys.path:
    sys.path.append(str(backend_path))

load_dotenv(backend_path / ".env")

BASE_URL = "http://localhost:8000"
DB_NAME = "teledoc"
MONGODB_URI = os.getenv("MONGODB_URI") or "mongodb://localhost:27017"

print(f"  API: {BASE_URL} |   DB: {DB_NAME}")
try:
    requests.get(f"{BASE_URL}/health", timeout=2)
    print("  Backend is ONLINE")
except:
    print("  Backend is OFFLINE. Please start it!")
```

```
API: http://localhost:8000 |   DB: teledoc
Backend is ONLINE
```

[2]:
```python
# LLM Judge Setup
try:
    from langchain_google_genai import ChatGoogleGenerativeAI
    if not os.getenv("GEMINI_API_KEY"):
        print("  GEMINI_API_KEY not found!")
        judge_llm = None
    else:
        judge_llm = ChatGoogleGenerativeAI(model="gemini-flash-latest",␣
 ↪google_api_key=os.getenv("GEMINI_API_KEY"))
        print("   Judge LLM Ready")
except ImportError:
    print("   LangChain not found. Scoring skipped.")
    judge_llm = None

def extract_json(content):
    content = str(content).strip()
    content = content.replace("```json", "").replace("```", "").strip()
    if "{" in content:
        start = content.find("{")
        end = content.rfind("}") + 1
        return content[start:end]
    return content

def evaluate_response(scenario, input_text, agent_response, expected_fact):
    if not judge_llm: return {"accuracy_score": 0.0, "reasoning": "No Judge LLM␣
 ↪or Missing Key"}

    prompt = f"""
    Role: Expert Medical AI Evaluator.
    Task: Score the AI's response.

    Scenario: {scenario}
    User Input: "{input_text}"
    Key Expected Fact: "{expected_fact}"
    AI Response: "{agent_response}"

    Evaluate:
    1. Did the AI correctly identify the visual content? (Binary)
    2. Is the answer helpful? (0.0-1.0)

    Return JSON ONLY:
    {{
        "fact_retrieved": boolean,
        "accuracy_score": float,
        "reasoning": "string"
    }}
    """
    try:
        response = judge_llm.invoke(prompt)
        content = response.content
        json_str = extract_json(content)

        try:
```

```
            parsed = json.loads(json_str)
        except json.JSONDecodeError:
            try:
                parsed = ast.literal_eval(json_str)
            except:
                return {"accuracy_score": 0.0, "reasoning": f"Parse Error:␣
↪{json_str[:50]}"}

        if isinstance(parsed, dict) and "accuracy_score" not in parsed and␣
↪"text" in parsed:
            inner = extract_json(parsed["text"])
            try:
                 parsed = json.loads(inner)
            except:
                 parsed = ast.literal_eval(inner)

        return parsed

    except Exception as e:
        return {"accuracy_score": 0.0, "reasoning": f"Eval Error: {str(e)}"}
```

```
 Judge LLM Ready
```

```
/Users/prateeksharma/Documents/TeleDoc/Evaluation/venv/lib/python3.14/site-
packages/langchain_core/_api/deprecation.py:26: UserWarning: Core Pydantic V1
functionality isn't compatible with Python 3.14 or greater.
  from pydantic.v1.fields import FieldInfo as FieldInfoV1
```

[3]:
```
# Helpers
AUTH_HEADERS = {}
CURRENT_PID = None

def login_and_get_pid(pid_prefix):
    global AUTH_HEADERS, CURRENT_PID
    email = f"{pid_prefix}@eval.com"
    try:
        r = requests.post(f"{BASE_URL}/auth/dev/login", json={"email": email},␣
↪timeout=10)
        if r.status_code != 200: return None
        token = r.json().get("access_token")
        AUTH_HEADERS = {"Authorization": f"Bearer {token}"}

        from pymongo import MongoClient
        client = MongoClient(MONGODB_URI)
        u = client[DB_NAME].users.find_one({"email": email})
        CURRENT_PID = u["patient_id"]
        return CURRENT_PID
    except: return None
```

```
def chat_interaction(msg, file_id=None, new_chat=True):
    try:
        if new_chat:
            r = requests.post(f"{BASE_URL}/agents/interaction/start",␣
 ↪headers=AUTH_HEADERS)
            global CID
            CID = r.json()["chat_id"]

        payload = {"message": msg}
        if file_id: payload["attachments"] = [file_id]

        r = requests.post(f"{BASE_URL}/agents/interaction/{CID}/message",␣
 ↪json=payload, headers=AUTH_HEADERS)
        return r.json()["reply"]
    except Exception as e:
        print(f"Chat Error: {e}")
        return ""

def reset_patient_data(pid, history=None, chat_summary=None, upload=None):
    from pymongo import MongoClient
    client = MongoClient(MONGODB_URI)
    db = client[DB_NAME]
    db.chats.delete_many({"patient_id": pid})
    db.reports.delete_many({"patient_id": pid})
    db.uploads.delete_many({"patient_id": pid})

    if history:
        db.medical_histories.update_one(
            {"patient_id": pid},
            {"$set": {"history": history}},
            upsert=True
        )
    else:
        # Default to avoid intake
        db.medical_histories.update_one(
            {"patient_id": pid},
            {"$set": {"history": "Established patient. No active complaints."}},
            upsert=True
        )

    if chat_summary:
         db.chats.insert_one({
            "chat_id": "past_chat_1",
            "patient_id": pid,
            "messages": [],
            "summary": chat_summary,
            "keywords": ["fracture", "finger", "pain"],
            "created_at": datetime(2023, 10, 1) # old date
        })

    if upload:
        # Insert text-only upload directly
         db.uploads.insert_one({
            "patient_id": pid,
            "filename": upload["filename"],
            "image_summary": upload["text"] # Using image_summary for text␣
↪content now
        })

def upload_real_file_vision(filename):
    path = Path(f"assets/{filename}")
    if not path.exists(): path = Path(f"Evaluation/assets/{filename}")
    if not path.exists():
        print(f"  File missing: {path}")
        return None

    with open(path, "rb") as f:
        r = requests.post(f"{BASE_URL}/patients/{CURRENT_PID}/uploads",
                          headers=AUTH_HEADERS, files={"file": (filename, f,␣
↪"image/png")})
        if r.status_code != 200:
            print(f"Upload failed: {r.text}")
            return None
        fid = r.json()["file_id"]

        # Wait for Vision
        from pymongo import MongoClient
        from bson import ObjectId
        client = MongoClient(MONGODB_URI)
        db = client[DB_NAME]
        print(f"  Vision Analyzing...", end="")
        for _ in range(25):
            doc = db.uploads.find_one({"file_id": ObjectId(fid)})
            if doc and doc.get("image_summary"):
                print(" Done!")
                return fid
            time.sleep(1)
            print(".", end="")
        print(" Timeout!")
        return fid
```

[4]:
```
# Run Full Evaluation (4 Scenarios)
results = []
```

```python
# 1. Diabetes
print("\n--- [1] Diabetes History ---")
if login_and_get_pid("eval_user_1"):
    reset_patient_data(CURRENT_PID, history='Patient has Type 2 Diabetes taking␣
↪Metformin')
    MSG = "I have a sore on my foot that isn't healing."
    ans = chat_interaction(MSG)
    score = evaluate_response("Diabetes Foot Ulcer", MSG, ans, "diabetes")
    results.append({"Scenario": "1. Medical History", "Score": score.
↪get("accuracy_score", 0.0)})

# 2. Fracture (Memory)
print("\n--- [2] Fracture Memory ---")
if login_and_get_pid("eval_user_2"):
    reset_patient_data(CURRENT_PID, chat_summary="Patient has a fracture of the␣
↪right index finger confirmed by X-Ray.")
    MSG = "My finger hurts again."
    ans = chat_interaction(MSG)
    score = evaluate_response("Previous Fracture Recall", MSG, ans, "fracture")
    results.append({"Scenario": "2. Memory Recall", "Score": score.
↪get("accuracy_score", 0.0)})

# 3. PDF (Text Analysis)
print("\n--- [3] PDF Analysis (Text) ---")
if login_and_get_pid("eval_user_3"):
    # We simulate a PDF text extraction by inserting directly into␣
↪image_summary (since we upgraded the field)
    reset_patient_data(CURRENT_PID, upload={"filename": "lab.pdf", "text":␣
↪"BLOOD TEST RESULTS... WBC: 15.0 (High)... Diagnosis: Infection likely."})
    MSG = "I have a fever, check my lab results."
    ans = chat_interaction(MSG)
    score = evaluate_response("High WBC Detection", MSG, ans, "15.0")
    results.append({"Scenario": "3. PDF Analysis", "Score": score.
↪get("accuracy_score", 0.0)})

# 4. Vision (Real Image)
print("\n--- [4] Vision (Real Image) ---")
if login_and_get_pid("eval_user_4"):
    reset_patient_data(CURRENT_PID) # Clean slate
    fid = upload_real_file_vision("chat_history.png")
    if fid:
        MSG = "What does conversation in this image say?"
        ans = chat_interaction(MSG, file_id=fid)
        score = evaluate_response("Vision OCR Test", MSG, ans, "chat interface␣
↪showing a conversation")
```

```
        results.append({"Scenario": "4. Vision Analysis", "Score": score.
 ↪get("accuracy_score", 0.0)})

# Final Report
print("\n" + "="*30)
df = pd.DataFrame(results)
print(df)
print("="*30)
df
```

```
--- [1] Diabetes History ---


--- [2] Fracture Memory ---


--- [3] PDF Analysis (Text) ---


--- [4] Vision (Real Image) ---

 Vision Analyzing…

.

.

. Done!

==============================
             Scenario  Score
0  1. Medical History    1.0
1    2. Memory Recall    1.0
2     3. PDF Analysis    1.0
3  4. Vision Analysis    1.0
==============================
```

[4]:

```
             Scenario  Score
0  1. Medical History    1.0
1    2. Memory Recall    1.0
2     3. PDF Analysis    1.0
3  4. Vision Analysis    1.0
```