

```

import pandas as pd
import json
import time
import requests
from google.colab import userdata
from sklearn.metrics import accuracy_score
from tqdm import tqdm

# Load dataset
df = pd.read_csv("/content/yelp.csv")[["text", "stars"]]

# Use a representative subset to avoid rate limits
df = df.sample(60, random_state=42).reset_index(drop=True)

df.head(), df.shape

(
              text  stars
0  We got here around midnight last Friday... the...      4
1  Brought a friend from Louisiana here. She say...      5
2  Every friday, my dad and I eat here. We order ...      3
3  My husband and I were really, really disappoin...      1
4  Love this place! Was in phoenix 3 weeks for w...      5,
(60, 2)

```

```

OPENROUTER_API_KEY = userdata.get("OPENROUTER_API_KEY")
print("API key loaded:", OPENROUTER_API_KEY[:8] + "...")

HEADERS = {
    "Authorization": f"Bearer {OPENROUTER_API_KEY}",
    "Content-Type": "application/json",
}

MODEL = "mistralai/mistral-7b-instruct"

```

API key loaded: sk-or-v1...

```

def call_llm(prompt: str) -> str:
    print("➡️ Calling LLM...")
    payload = {
        "model": MODEL,
        "messages": [{"role": "user", "content": prompt}],
        "temperature": 0.2,
        "max_tokens": 200
    }

    r = requests.post(
        "https://openrouter.ai/api/v1/chat/completions",
        headers=HEADERS,
        data=json.dumps(payload),
        timeout=30
    )

    print("➡️ Status:", r.status_code)

    if r.status_code != 200:
        print("❌ Error:", r.text)
        r.raise_for_status()

    return r.json()["choices"][0]["message"]["content"].strip()

```

```

print(
    call_llm(
        'Return ONLY valid JSON: {"predicted_stars":5,"explanation":"Great"}'
    )
)

```

```

➡️ Calling LLM...
➡️ Status: 200
{"predicted_stars":5,"explanation":"Great"}

```

Prompts

```

def prompt_v1(text):
    return f"""

```

```
Classify the Yelp review into a star rating from 1 to 5.  
Return ONLY valid JSON:  
{ {"predicted_stars": <1-5>, "explanation": "<short>"}}
```

Review:
"{}"
""

```
def prompt_v2(text):  
    return f"""  
Use this rubric:  
1 = extremely negative  
2 = mostly negative  
3 = mixed  
4 = mostly positive  
5 = extremely positive  
  
Return ONLY valid JSON:  
{ {"predicted_stars": <1-5>, "explanation": "<one sentence>"}}
```

Review:
"{}"
""

```
def prompt_v3(text):  
    return f"""  
Step 1: assess sentiment  
Step 2: map to Yelp stars  
Step 3: validate consistency  
  
Return ONLY valid JSON:  
{ {"predicted_stars": <1-5>, "explanation": "<brief>"}}
```

Review:
"{}"
""

```
def parse_json(txt):  
    try:  
        j = json.loads(txt)  
        if isinstance(j.get("predicted_stars"), int):  
            return j["predicted_stars"], True  
    except:  
        pass  
    return None, False
```

```
def run_experiment(prompt_fn, label):  
    print(f"\n===== RUNNING {label} =====")  
    preds, valids = [], []  
  
    for i, row in df.iterrows():  
        print(f"\nReview {i+1}/{len(df)}")  
        out = call_llm(prompt_fn(row["text"]))  
        print("Raw:", out)  
  
        pred, ok = parse_json(out)  
        preds.append(pred)  
        valids.append(ok)  
  
        time.sleep(0.6) # rate-limit safe  
  
    clean = df.copy()  
    clean["pred"] = preds  
    clean = clean.dropna()  
  
    acc = accuracy_score(clean["stars"], clean["pred"])  
    json_rate = sum(valids) / len(valids)  
  
    print(f"{label} → Accuracy: {acc:.3f}, JSON rate: {json_rate:.2%}")  
    return acc, json_rate
```

```
results = []  
  
results.append(("Baseline", *run_experiment(prompt_v1, "Prompt V1")))
```

```
results.append(("Rubric", *run_experiment(prompt_v2, "Prompt V2")))
results.append(("Self-Validated", *run_experiment(prompt_v3, "Prompt V3")))
```

===== RUNNING Prompt V1 =====

Review 1/60
+ Calling LLM...
- Status: 200
Raw:

Review 2/60
+ Calling LLM...
- Status: 200
Raw:

Review 3/60
+ Calling LLM...
- Status: 200
Raw:

Review 4/60
+ Calling LLM...
- Status: 200
Raw:

Review 5/60
+ Calling LLM...
- Status: 200
Raw:

Review 6/60
+ Calling LLM...
- Status: 200
Raw:

Review 7/60
+ Calling LLM...
- Status: 200
Raw:

Review 8/60
+ Calling LLM...
- Status: 200
Raw:

Review 9/60
+ Calling LLM...
- Status: 200
Raw: <s> {"predicted_stars": 5, "explanation": "The reviewer praises the place for its great ambiance, generous portions,

Review 10/60
+ Calling LLM...
- Status: 200
Raw:

Review 11/60
+ Calling LLM...
- Status: 200
Raw: {"predicted_stars": 4, "explanation": "The review mentions positive aspects such as good food, ability to handle a la

```
comparison = pd.DataFrame(
    results,
    columns=["Prompt Strategy", "Accuracy", "JSON Validity Rate"]
)

comparison
```

	Prompt Strategy	Accuracy	JSON Validity Rate
0	Baseline	0.666667	0.05
1	Rubric	0.533333	0.25
2	Self-Validated	0.666667	0.05

Start coding or [generate](#) with AI.

