Team 1 Project: AI CS Tutor

CSC 659-859
Summer 2025
Harris Chan, Maeve Fitzpatrick, Zari Haidarian

Summary

Purpose/Goal of Report: To determine if GenAI (specifically GPT-4) can reliably grade and explain answers to multiple choice questions

Data Used: Multiple-choice questions gathered from a GitHub repository

Data: GitHub Questions

We wanted questions that were:

- simply worded
- had answers that were short (one word or only a few words)
- didn’t require examples

It’s Copyright–free!

About

This is the repository to help Computer Science students By Making All type of MCQ Questions at the place

hacktoberfest hactoberfest-accepted hactoberfest2022

Readme
MIT license
Activity
5 stars
1 watching
18 forks

Report repository

Training DB audit Checklist

1. **How are feature (variable) data obtained and their meaning:**

   Each MCQ Feature consists of the following

- Question_Stem (text)
- Options A–D (text)
- Correct_Option (categorical label)

2. **How are class labels obtained/verified wrt. ground truth:**

   Each MCQ's correct answer is verified from the MCQ answer key. Answer keys are created by humans as well as reviewed by humans for reliability.

3. **Is demography well covered in adequate and fair way:**

- Dataset contains questions with definitive answers
- Questions are evenly represented across multiple computer science topics.

4. **Number of samples in each class; is the data unbalanced:**

   Classes: Options A, B, C, D
   Each of the four labels will have relatively balanced frequency. No class will have less than 10% of all class samples to ensure balanced data.

Training DB audit Checklist

5. **Type of features (numerical, categorical nominal or ordinal):**

   Question_Stem and Options: Categorical Nominal

   Correct_Option: Categorical Nominal

6. **Missing values:**

   There are no missing values

7. **Are there enough samples compared to number of features used:**

   Features: 5 per question

   Samples: 50 questions

   Ratio: 5 * 10 = 50

   It meets the requirements

8. **List and description of features, formats are well documented:**

   All fields (Question_Stem, Option_A–D, Correct_Option) are documented and stored in a structured CSV.

9. **Check privacy issues (no personal features):**

   The dataset contains no personal identifiers or user inputs. The dataset is ok for classroom use

Application Overview

Our prototype is an Automated MCQ Answerer built entirely in a Jupyter notebook. It:

1. Loads one question at a time (stem + four options) from a CSV
2. Constructs a chat prompt for GPT-4, including a system message, a few-shot template, and the actual question
3. Calls the OpenAI API (model gpt-4o-mini)
4. Parses the model's reply to extract the chosen option letter and explanation
5. Displays results with accuracy % and Confusion Matrix

GenAI Environment

- Model: GPT-4 series (gpt-4o-mini endpoint)
- SDK: OpenAI Python SDK v0.27+
- Runtime: Python 3.10 within JupyterLab (CPU only)
- Dependencies: openai, pandas, numpy, matplotlib (for confusion-matrix plotting)

Prompt Customization – System Message

System Message

You are an expert computer-science tutor. Given a multiple-choice question with four options, select the correct answer and provide a one-sentence rationale in clear, beginner-friendly language.

```python
def query_gpt4_mcq(question_stem, options_dict, few_shot=True):
    system_message = (
        "You are an expert computer-science tutor. Given a multiple-choice question with four options, "
        "select the correct answer and provide a one-sentence rationale in clear, beginner-friendly language. "
        "Respond exactly in this format:\n\n"
        "Answer: <letter>) <option text>\n"
        "Explanation: <one-sentence rationale>"
    )
```

Prompt Customization - Few-Shot Examples

```
prompt = [
  {"role": "system", "content": system_message},

  # Example 1
  {"role":"user", "content":
    "Q: Which data structure follows FIFO?\n"
    "A) Stack   B) Queue   C) Tree   D) Graph"
  },
  {"role":"assistant", "content":
    "Answer: B) Queue\n"
    "Explanation: A queue enqueues and dequeues elements in first-in, first-out
order."
  },
```

Screenshots of our code (1/3)

Step 1: Install dependencies

```
[ ]
# Install dependencies
!pip install --quiet openai pandas numpy matplotlib scikit-learn sentence-transformers
```

Step 2: Upload the CSV

```
[ ]
from google.colab import files
import pandas as pd

# Upload the CSV (choose manual_50_questions.csv from your desktop)
uploaded = files.upload()  # interactive picker
df = pd.read_csv("test_50_questions.csv")
print(f"Loaded {len(df)} questions")
df.head(2)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the cu

```
Saving test_50_questions.csv to test_50_questions (5).csv
Loaded 50 questions
```

	question_id	topic	stem	option_A	option_B
0	Q01	AI	What is the full form of AI?	Artificially Intelligent	Artificial Intelligence
1	Q02	AI	What is Artificial Intelligence?	Artificial Intelligence is a field that aims t...	Artificial Intelligence is a field that aims t...

Step 3: Set the OpenAI API key securely

```
[ ]
import os
from getpass import getpass
from openai import OpenAI

# Securely input API key (Colab will hide input)
os.environ["OPENAI_API_KEY"] = getpass("Enter your OpenAI API key: ")
client = OpenAI()  # picks up the key from the environment
```

```
Enter your OpenAI API key: ······
```

Screenshots of Our Code (2/3)

Step 4: Define the query function (few-shot MCQ answerer)

```python
import re

def query_gpt4_mcq(question_stem, options_dict, few_shot=True):
    system_message = (
        "You are an expert computer-science tutor. Given a multiple-choice question with four options, "
        "select the correct answer and provide a one-sentence rationale in clear, beginner-friendly language. "
        "Respond exactly in this format:\n\n"
        "Answer: <letter>) <option text>\n"
        "Explanation: <one-sentence rationale>"
    )

    messages = [{"role": "system", "content": system_message}]
    if few_shot:
        # Example 1
        messages.append({
            "role": "user",
            "content": "Q: Which data structure follows FIFO?\nA) Stack   B) Queue   C) Tree   D) Graph"
        })
        messages.append({
            "role": "assistant",
            "content": "Answer: B) Queue\nExplanation: A queue enqueues and dequeues in first-in, first-out order."
        })
        # Example 2
        messages.append({
            "role": "user",
            "content": "Q: What is the worst-case time complexity of bubble sort?\nA) O(n)   B) O(n log n)   C) O(n²)   D) O(log n)"
        })
        messages.append({
            "role": "assistant",
            "content": "Answer: C) O(n²)\nExplanation: Bubble sort compares adjacent elements repeatedly, giving O(n²) in the worst case."
        })

    q_text = f"Q: {question_stem}\n"
    q_text += f"A) {options_dict['A']}   B) {options_dict['B']}   C) {options_dict['C']}   D) {options_dict['D']}"
    messages.append({"role": "user", "content": q_text})

    # New API call
    resp = client.chat.completions.create(
        model="gpt-4o-mini",  # or "gpt-4" if available/preferred
        messages=messages,
        temperature=0.2,
        max_tokens=150,
    )
    raw = resp.choices[0].message.content.strip()

    # Parse predicted option letter
    answer_match = re.search(r"Answer:\s*([A-D])\)", raw)
    predicted_option = answer_match.group(1) if answer_match else None

    # Parse explanation
    expl_match = re.search(r"Explanation:\s*(.+)", raw, re.DOTALL)
    explanation = expl_match.group(1).strip() if expl_match else ""
    if explanation:
        explanation = explanation.split(".")[0].strip() + "."

    return predicted_option, explanation, raw
```

Step 5: Batch run over the 50 questions and log results

```python
import datetime

results = []
for _, row in df.iterrows():
    opts = {
        "A": row["option_A"],
        "B": row["option_B"],
        "C": row["option_C"],
        "D": row["option_D"]
    }
    predicted, explanation, raw = query_gpt4_mcq(row["stem"], opts, few_shot=True)
    results.append({
        "question_id": row["question_id"],
        "topic": row["topic"],
        "stem": row["stem"],
        "correct_option": row["correct_option"],
        "predicted_option": predicted,
        "explanation": explanation,
        "reference_explanation": row.get("reference_explanation",""),
        "raw_model_output": raw,
        "timestamp": datetime.datetime.utcnow().isoformat()
    })

results_df = pd.DataFrame(results)
results_df.to_csv("model_outputs_batch.csv", index=False)
results_df.head()
```

	question_id	topic	stem	correct_option	predicted_option	explanation
0	Q01	AI	What is the full form of AI?	B	B	AI stands for Artificial Intelligence, which r...
1	Q02	AI	What is Artificial Intelligence?	C	C	Artificial Intelligence focuses on creating ma...
2	Q03	AI	Who is the inventor of Artificial Intelligence?	C	C	John McCarthy is credited with coining the ter...
3	Q04	AI	Which of the following is the branch of Artifi...	A	A	Machine Learning is a branch of Artificial Int...
4	Q05	AI	What is the goal of Artificial Intelligence?	C	D	The main goal of Artificial Intelligence is to...

Screenshots of Our Code (3/3)

Step 6: Compute accuracy and confusion matrix

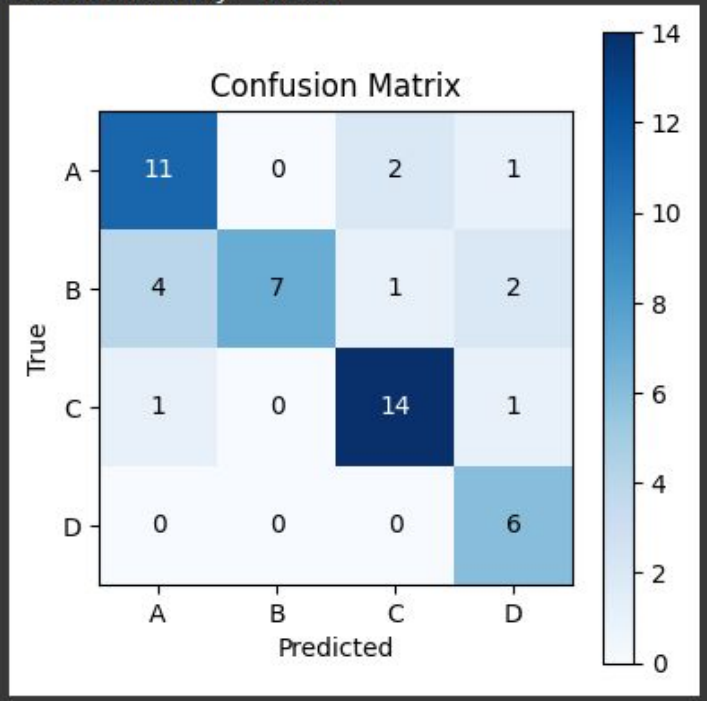
```python
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

clean = results_df.dropna(subset=["predicted_option"])
y_true = clean["correct_option"].tolist()
y_pred = clean["predicted_option"].tolist()
labels = ['A','B','C','D']

acc = accuracy_score(y_true, y_pred)
print(f"Overall accuracy: {acc:.2%}")

cm = confusion_matrix(y_true, y_pred, labels=labels)
fig, ax = plt.subplots(figsize=(4,4))
im = ax.imshow(cm, interpolation='nearest', cmap='Blues')
ax.set_xticks(np.arange(len(labels))); ax.set_yticks(np.arange(len(labels)))
ax.set_xticklabels(labels); ax.set_yticklabels(labels)
ax.set_xlabel("Predicted"); ax.set_ylabel("True")
ax.set_title("Confusion Matrix")
for i in range(len(labels)):
    for j in range(len(labels)):
        ax.text(j, i, cm[i,j], ha='center', va='center',
                color='white' if cm[i,j] > cm.max()/2 else 'black')
fig.colorbar(im, ax=ax)
plt.tight_layout()
plt.show()
```

```
Overall accuracy: 76.00%
```



Overall Success Rate

Out of 50 questions,

the AI got **38 correct → 76% accuracy**

Domain	# Questions	# Correct	Accuracy
Data Structures	10	8	80%
Algorithms	10	7	70%
Machine Learning	10	8	80%
Databases & SQL	10	9	90%
CS Foundations	10	6	60%

	Predicted A	Predicted B	Predicted C	Predicted D	Row Total
Actual A	10	0	2	1	13
Actual B	4	7	1	2	14
Actual C	1	0	14	1	16
Actual D	0	0	0	7	7
Col Total	15	7	17	11	50

Error Results

Question	Correct Answer	Model Prediction	Model Explanation	Error Type	Why It Matters
The worst thing that missing data does is lower sample size and reduce power. (1) TRUE (2) FALSE	2	1	“Missing data can lead to a smaller sample size, which reduces the statistical power of the analysis.”	Plausible-but-wrong	There are worse consequences of missing data than just reduced power (ex: introducing bias, causing systematic errors). The model oversimplified the impact of missing data by focusing only on the most obvious consequence.

Error Results

Question	Correct Answer	Model Prediction	Model Explanation	Error Type	Why It Matters
For two variables, X and Y, there can be a maximum of ___ lines (1) One (2) Two (3) Three (4) Four	1	3	“With two variables, you can create three possible lines: one for each variable alone and one for their combination.”	Misinterpretation	It misunderstood what is a "regression line" between two variables. In standard regression analysis, there is one best-fit line to describe the relationship between X and Y variables. The model incorrectly considered separate univariate scenarios rather than focusing on the bivariate relationship, showing confusion about fundamental regression concepts.

Audit for Ethics & Trustworthiness (Model-Card Style)

The Model Card is a concise document that describes a machine-learning model's intended use, performance metrics, and ethical considerations under standardized headings. It covers:

- Intended Use & Limitations
- Performance Metrics (accuracy, per-slice eval)
- Ethical Considerations (bias, fairness, privacy, safety)
- Maintenance & Monitoring

Reference: Mitchell, M. et al. "Model Cards for Model Reporting," FAT 2019. https://arxiv.org/pdf/1810.03993

Audit Results, Model-Card Style

Dimension	Assessment
Accuracy & Error Risk	Measured overall accuracy of 76.0% on 50 held-out MCQs. Confusion matrix reveals most confusions between semantically close distractors.
Bias & Fairness	No demographic or sensitive group features in data. Topic coverage audited to ensure balanced representation across subdomains (e.g., data structures, algorithms, databases, ML, stats).
Explainability & Transparency	Each prediction is accompanied by a one-sentence rationale. We categorize error types (misinterpretation, plausible-but-wrong, hallucination) to surface failure modes.
Human Oversight & Control	All model outputs are reviewed by a human before inclusion in the report. Misclassifications and explanations flagged for manual correction.
Privacy & Security	No user data is ingested at inference time—only public CS question text. The model runs in a closed environment with no external logging of user identifiers.
Environmental & Social Impact	Negligible compute footprint (single batch of 50 queries). Outputs used solely for educational audit; no high-stakes deployment.
Accountability & Governance	We version-control the evaluation code and CSV; all results reproducible via provided notebook. Team members rotate audit roles to reduce individual bias.

Model Confidence & Recommendations

We viewed the AI's internal confidence scores (token-probability gaps) as a guide: high-confidence answers were often correct, whereas low-confidence ones flagged areas needing review. However, given the 76% overall accuracy, we treat confidence as a sanity check rather than a replacement for human judgment.

Recommendations

1. Use the AI Tutor as an Assistive Tool, Not a Standalone
   - Leverage it to generate quick answer drafts and concise rationales.
   - Always cross-check with official answer keys or subject-matter experts before finalizing.
2. Integrate Human-in-the-Loop Review
   - Flag all low-confidence and "plausible-but-wrong" cases for human verification.
   - Embed a simple review interface allowing instructors to accept, edit, or reject model suggestions.
3. Continual Monitoring & Retraining
   - Periodically test new question sets to detect drifts in topic accuracy.
   - Fine-tune or prompt-engineer further on the lower-scoring bucket ("Misc. CS Foundations") to boost performance.
4. Transparency & Documentation
   - Maintain a living audit log of all model outputs, confidence scores, and human corrections.
   - Share this documentation with stakeholders to build trust and enable reproducibility.

Unresolved Issues & Conclusion

Unresolved Issues

- Accuracy Ceiling: At 76%, the model still misclassified ~1 in 4 questions—unacceptable for high-stakes assessments.
- Overconfidence Risk: Even high-confidence wrong answers can mislead; sole reliance is risky.
- Explainability Limits: Single-sentence rationales may oversimplify complex concepts and obscure subtle misunderstandings.

Conclusion:
We recommend adopting the AI tutor as a supplemental learning aid—to speed up feedback cycles and spark student engagement—but not as an authoritative grader. By combining the model's rapid responses with structured human review and ongoing audits, we can harness its strengths while mitigating its current limitations.

Any question?

Thank you